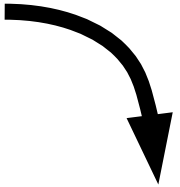
Machine Learning with scikit-learn

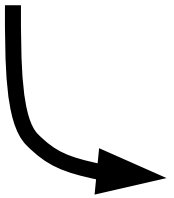
Andreas Mueller (NYU Center for Data Science, co-release manager scikit-learn)

Me


universitätbonn
ais




Amazon
Machine
Learning




NYU
Center for
Data Science

What is scikit-learn?

Classification
Regression
Clustering
Semi-Supervised Learning
Feature Selection
Feature Extraction
Manifold Learning
Dimensionality Reduction
Kernel Approximation
Hyperparameter Optimization
Evaluation Metrics
Out-of-core learning
……

Home Installation Documentation Examples

Google™ Custom Search Search ×

Documentation of scikit-learn 0.16-git

Quick Start

A very short introduction into machine learning problems and how to solve them using scikit-learn. Introduced basic concepts and conventions.

User Guide

The main documentation. This contains an in-depth description of all algorithms and how to apply them.

Other Versions

- scikit-learn 0.15 (stable)
- scikit-learn 0.16 (development)
- scikit-learn 0.14
- scikit-learn 0.13
- scikit-learn 0.12
- Older versions ▲

Tutorials

Useful tutorials for developing a feel for some of scikit-learn's applications in the machine learning field.

API

The exact API of all functions and classes, as given by the docstrings. The API documents expected types and allowed features for all functions, and all parameters available for the algorithms.

Additional Resources

Talks given, slide-sets and other information relevant to scikit-learn.

Contributing

Information on how to contribute. This also contains useful information for advanced users, for example how to build their own estimators.

Flow Chart

A graphical overview of basic areas of machine learning, and guidance which kind of algorithms to use in a given situation.

FAQ

Frequently asked questions about the project and contributing.

http://scikit-learn.org/

What is machine learning?

Hi Andy,

I just received an email from the first tutorial speaker, presenting right before you, saying he's ill and won't be able to make it.

I know you have already committed yourself to two presentations, but is there anyway you could increase your tutorial time slot, maybe just offer time to try out what you've taught? Otherwise I have to do some kind of modern dance interpretation of Python in data :-)
-Leah

Hi Andreas,

I am very interested in your Machine Learning background. I work for X Recruiting who have been engaged by Z, a worldwide leading supplier of Y. We are expanding the core engineering team and we are looking for really passionate engineers who want to create their own story and help millions of people.

Can we find a time for a call to chat for a few minutes about this?

Thanks

Hi Andy,

I just received an email fr the first tutorial speaker, presenting , saying he's ill and won't be

I know you have a urself to two presentations you could increase you , maybe just offer time to try out what you've taught? Otherwise I have to do some kind of modern dance interpretation of Python in data :-)
-Leah

Hi Andreas,

I am very interested in your Machine Learning background. I work for X Recruiting who have been engaged by Z, a worldwide leading supplier of Y. We are expanding the core engineering team and we are looking for really passionate engineers who own story and help millions of

Can we find a ti for a few minutes about t

Thanks

Doing Machine Learning With Scikit-Learn

Representing Data

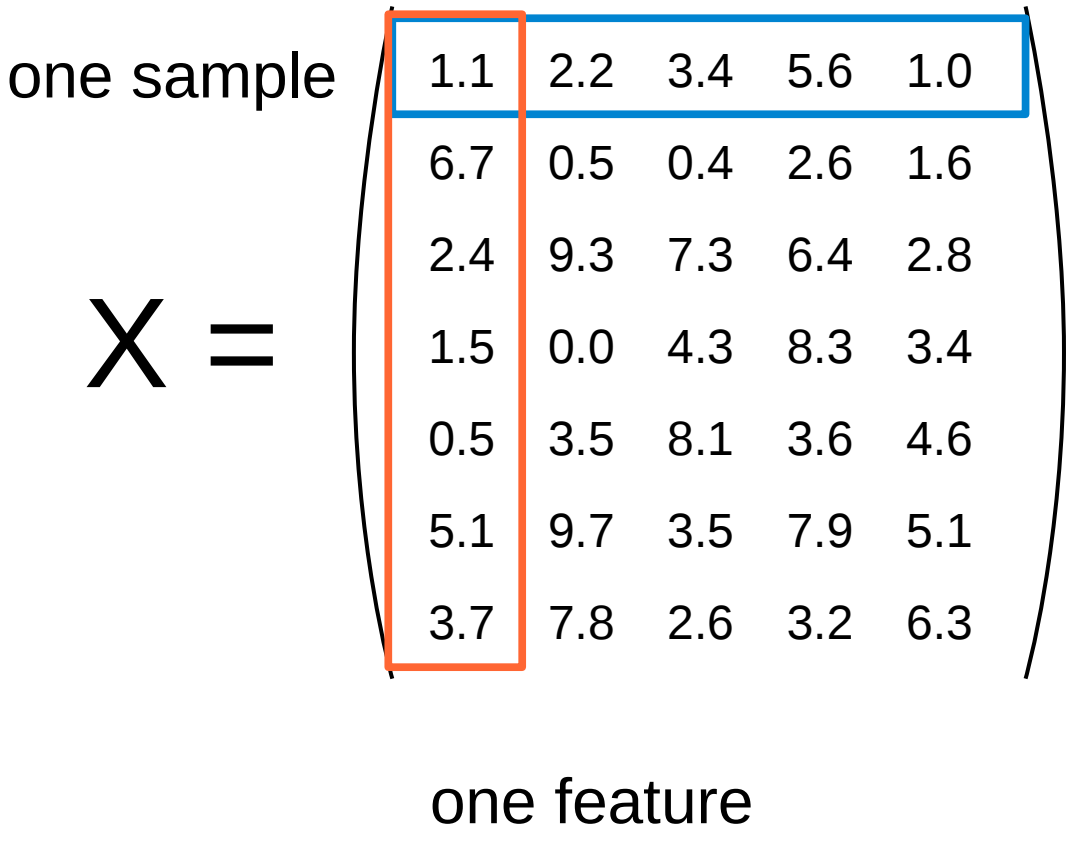
$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix}$$

Representing Data

one sample

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix}$$

Representing Data



Representing Data

one sample

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix}$$

one feature

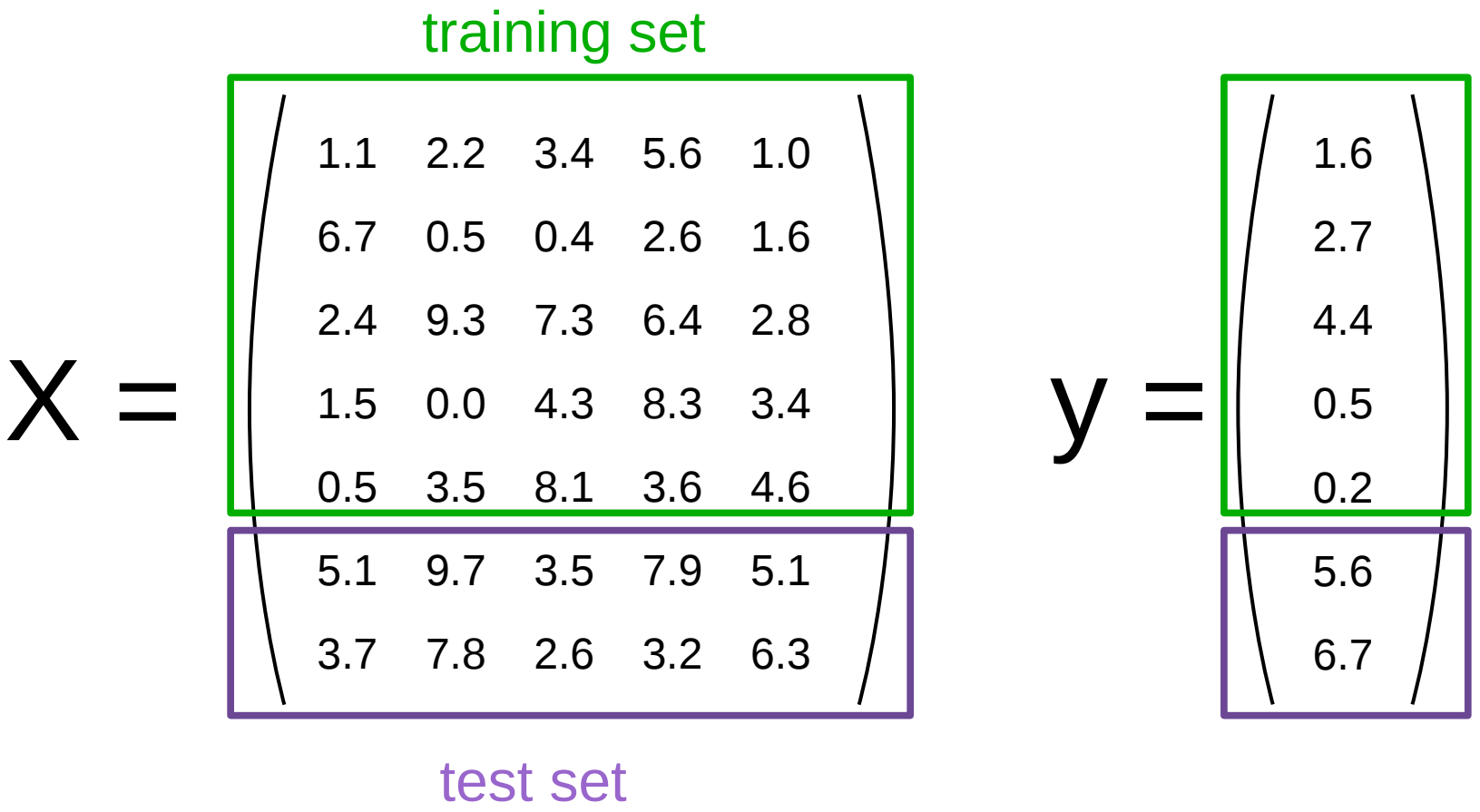
$$y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

outputs / labels

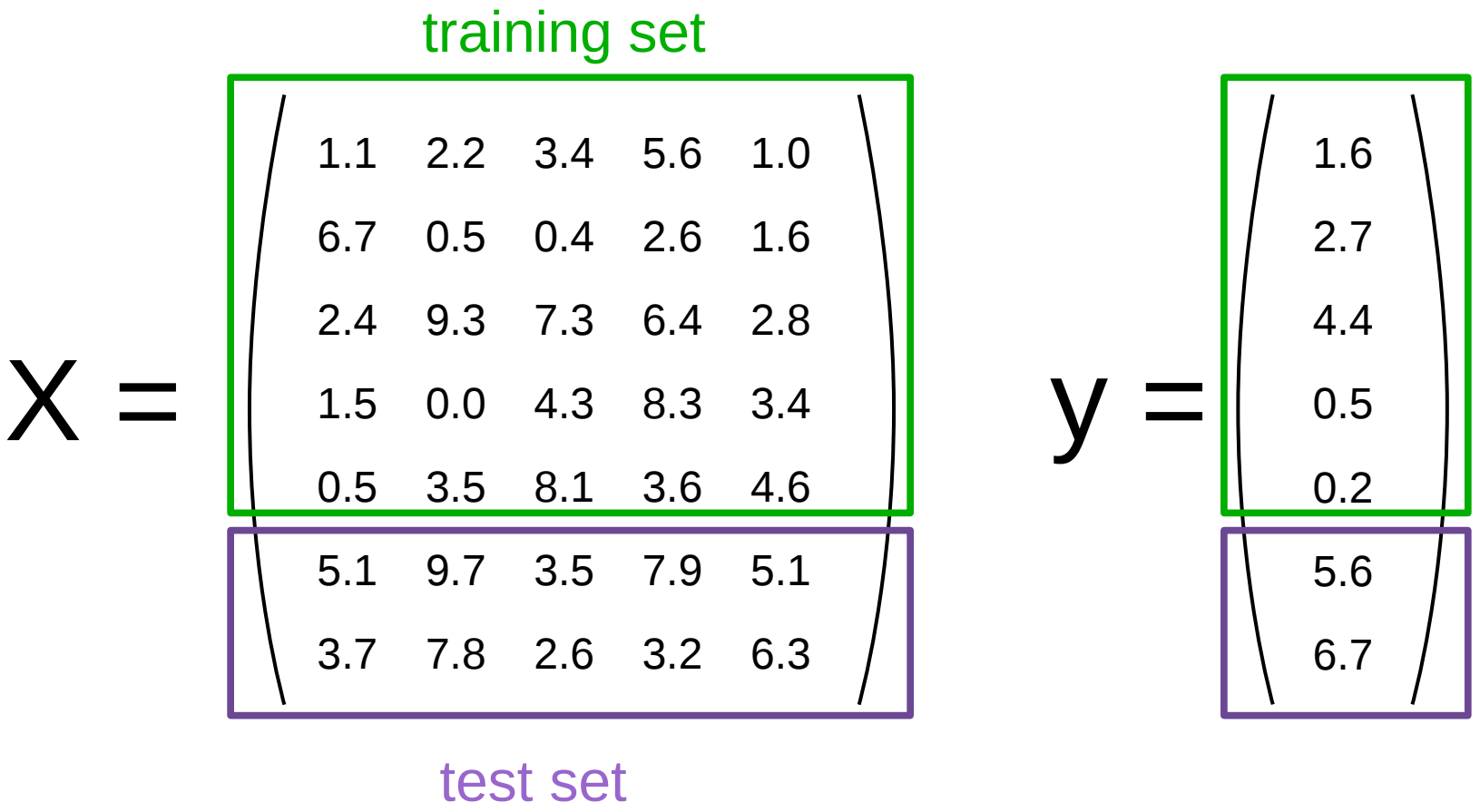
Training and Testing Data

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix} \quad y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

Training and Testing Data

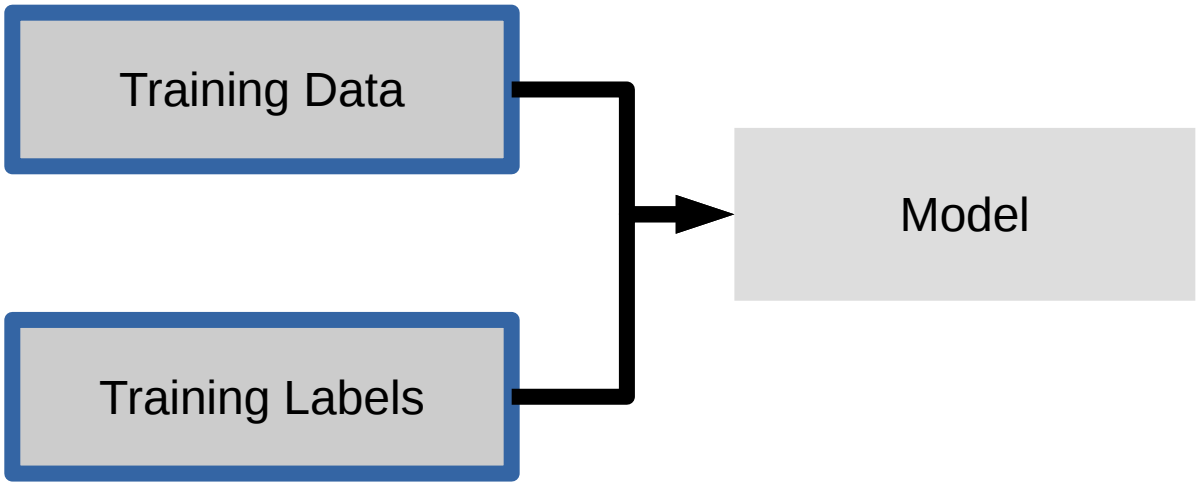


Training and Testing Data

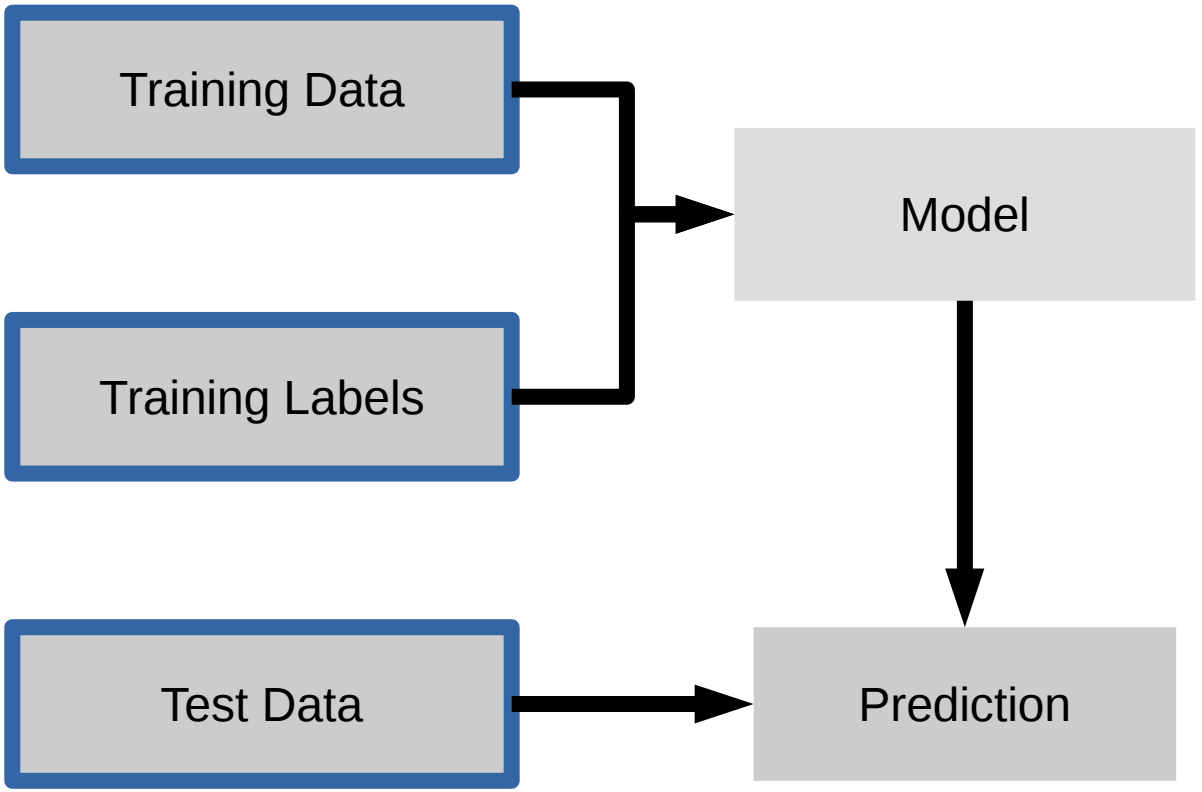



```
from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

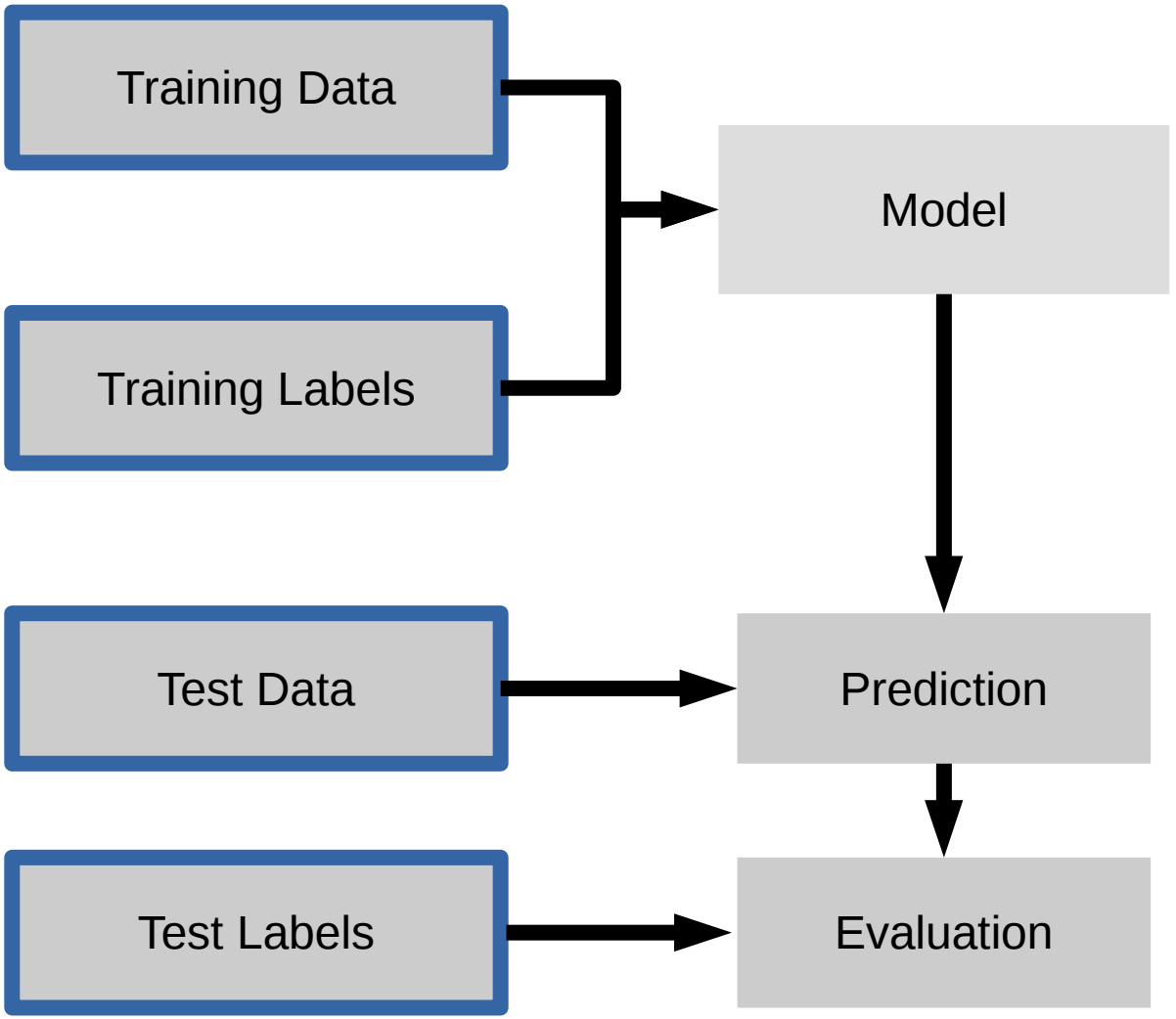
Supervised Machine Learning



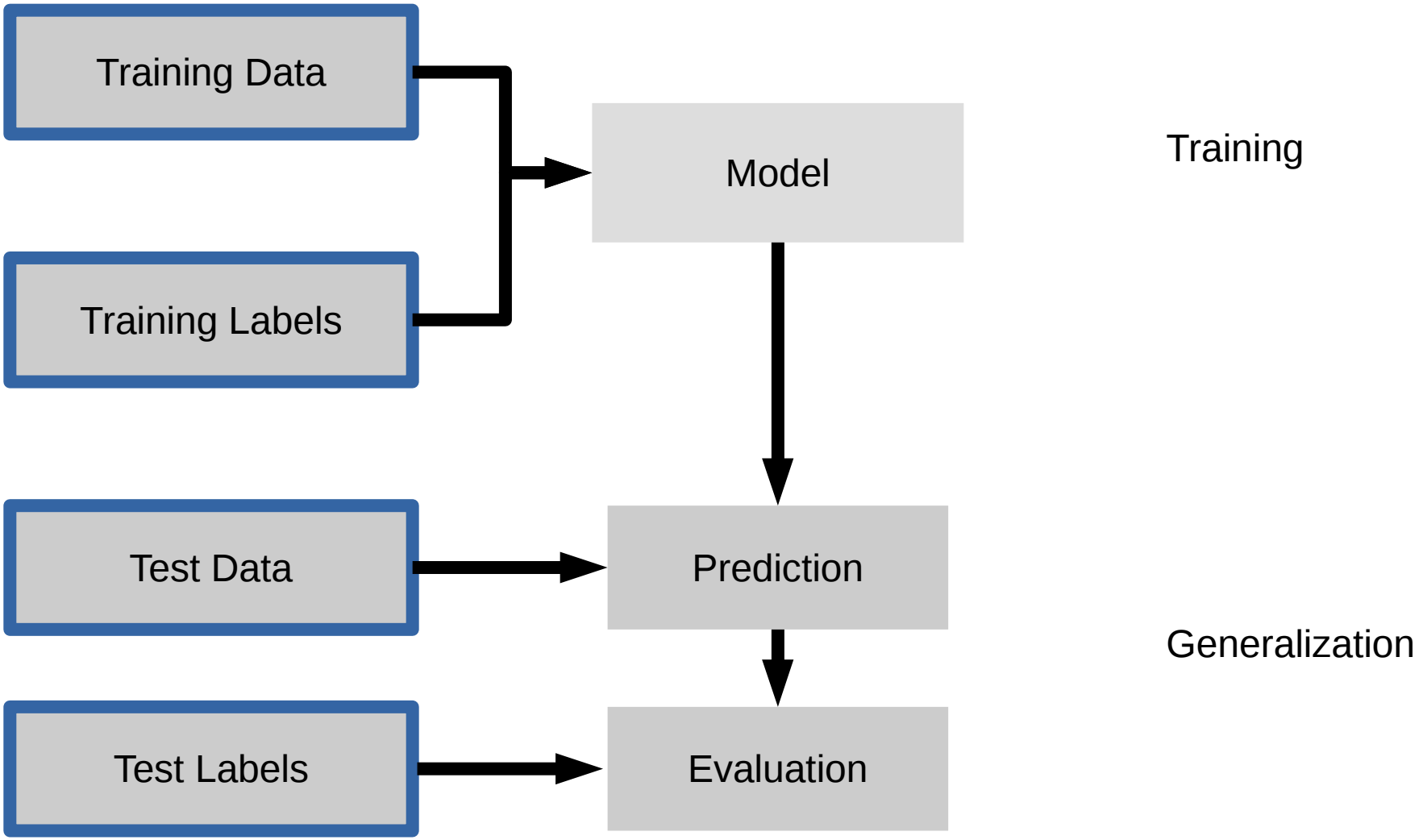
Supervised Machine Learning



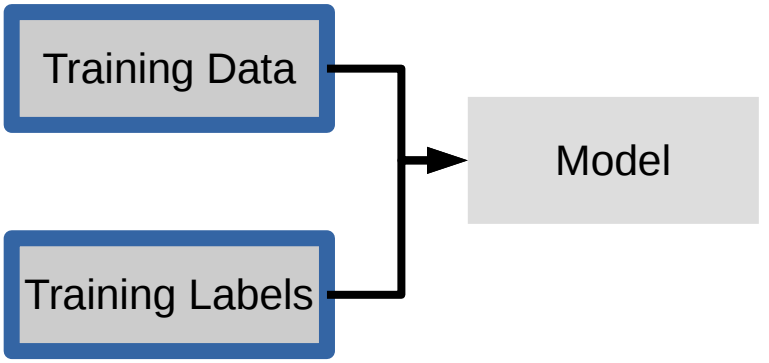
Supervised Machine Learning



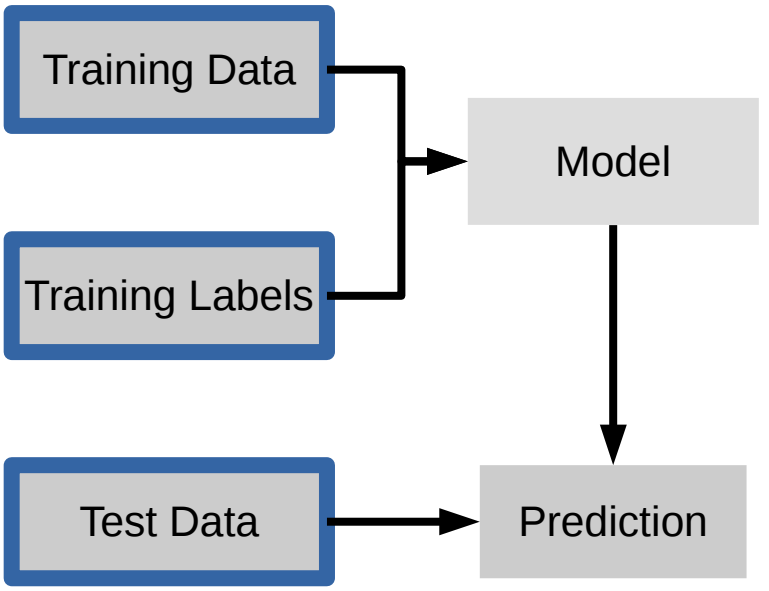
Supervised Machine Learning



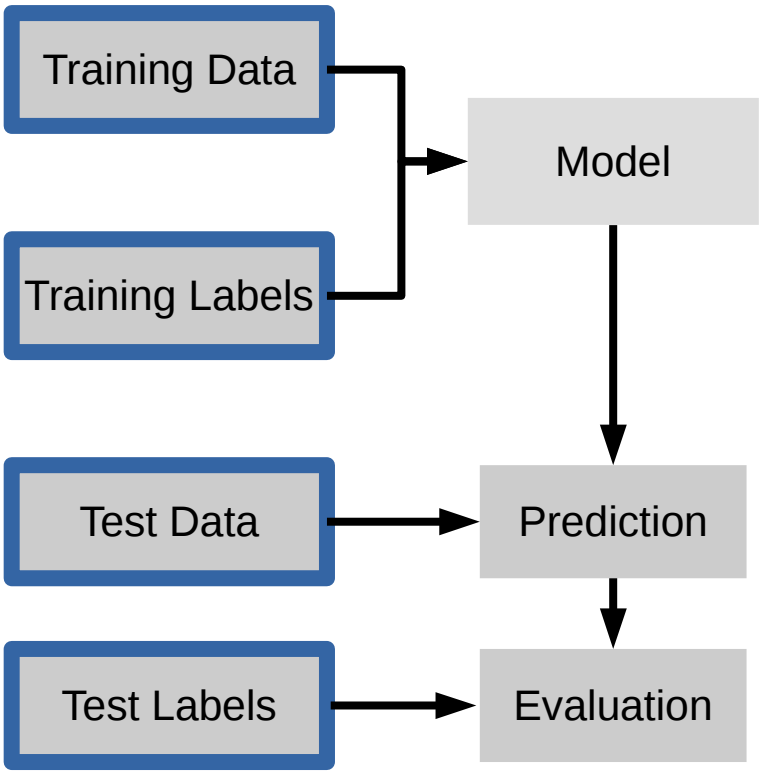
```
clf = RandomForestClassifier()

clf.fit(X_train, y_train)
```



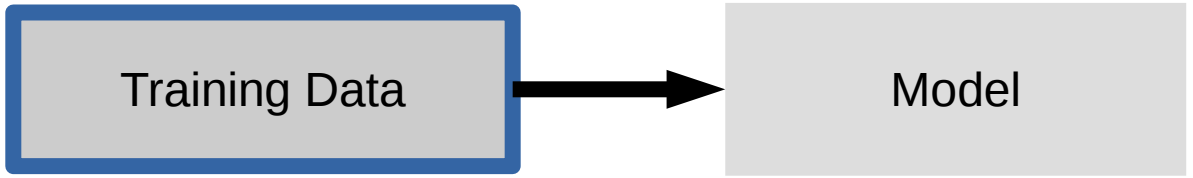
```
clf = RandomForestClassifier()

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```



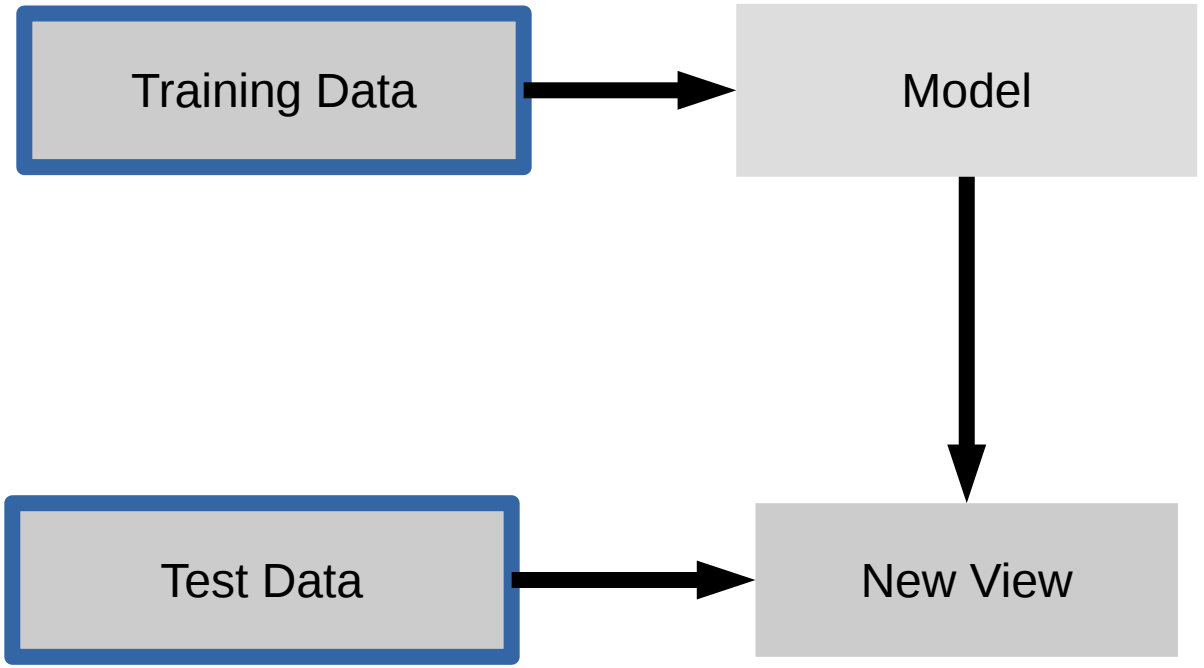
```
clf = RandomForestClassifier()

clf.fit(X_train, y_train)


y_pred = clf.predict(X_test)

clf.score(X_test, y_test)
```



Unsupervised Machine Learning



Unsupervised Machine Learning

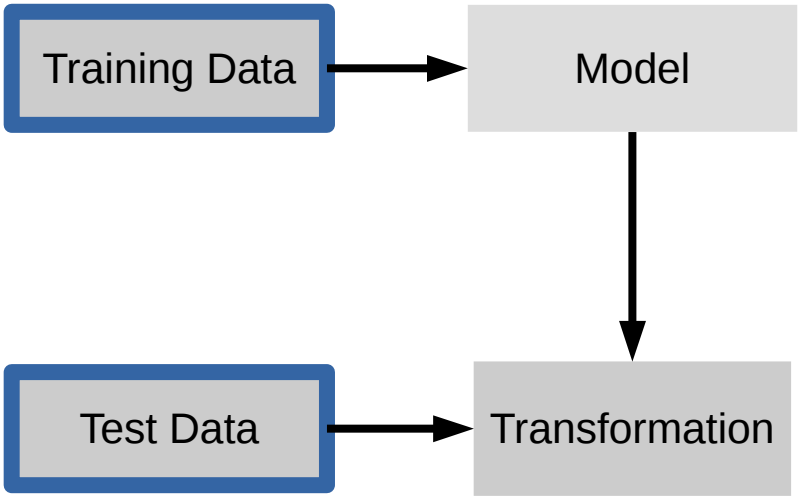


Unsupervised Transformations

```
pca = PCA()
```

```
pca.fit(X_train)
```

```
X_new = pca.transform(X_test)
```



Basic API

```
estimator.fit(X, [y])
```

`estimator.predict`	`estimator.transform`
Classification	Preprocessing
Regression	Dimensionality reduction
Clustering	Feature selection
	Feature extraction

Sample application: Sentiment Analysis

IMDB Movie Reviews Data

Review:
One of the worst movies I've ever rented. Sorry it had one of my favorite actors on it (Travolta) in a nonsense role. In fact, anything made sense in this movie.

Who can say there was true love between Eddy and Maureen? Don't you remember the beginning of the movie ?

Is she so lovely? Ask her daughters. I don't think so.

Label: negative

Training data: 12500 positive, 12500 negative

Bag Of Word Representations

`CountVectorizer / TfidfVectorizer`

Bag Of Word Representations

CountVectorizer / TfidfVectorizer

“This is how you get ants.”

Bag Of Word Representations

`CountVectorizer / TfidfVectorizer`

`“This is how you get ants.”`

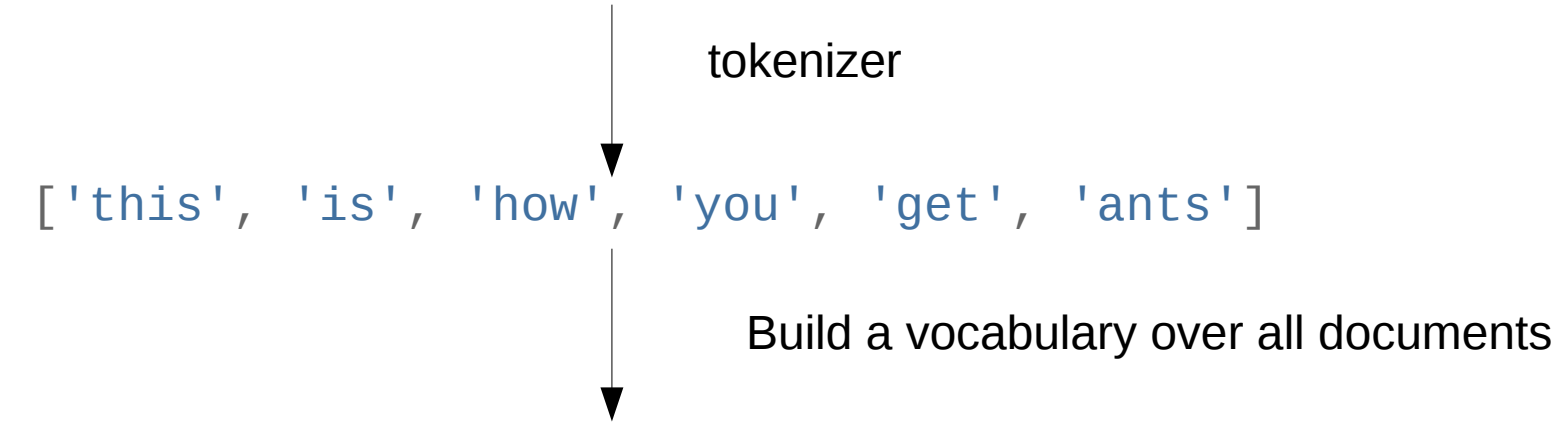
tokenizer

```
['this', 'is', 'how', 'you', 'get', 'ants']
```

Bag Of Word Representations

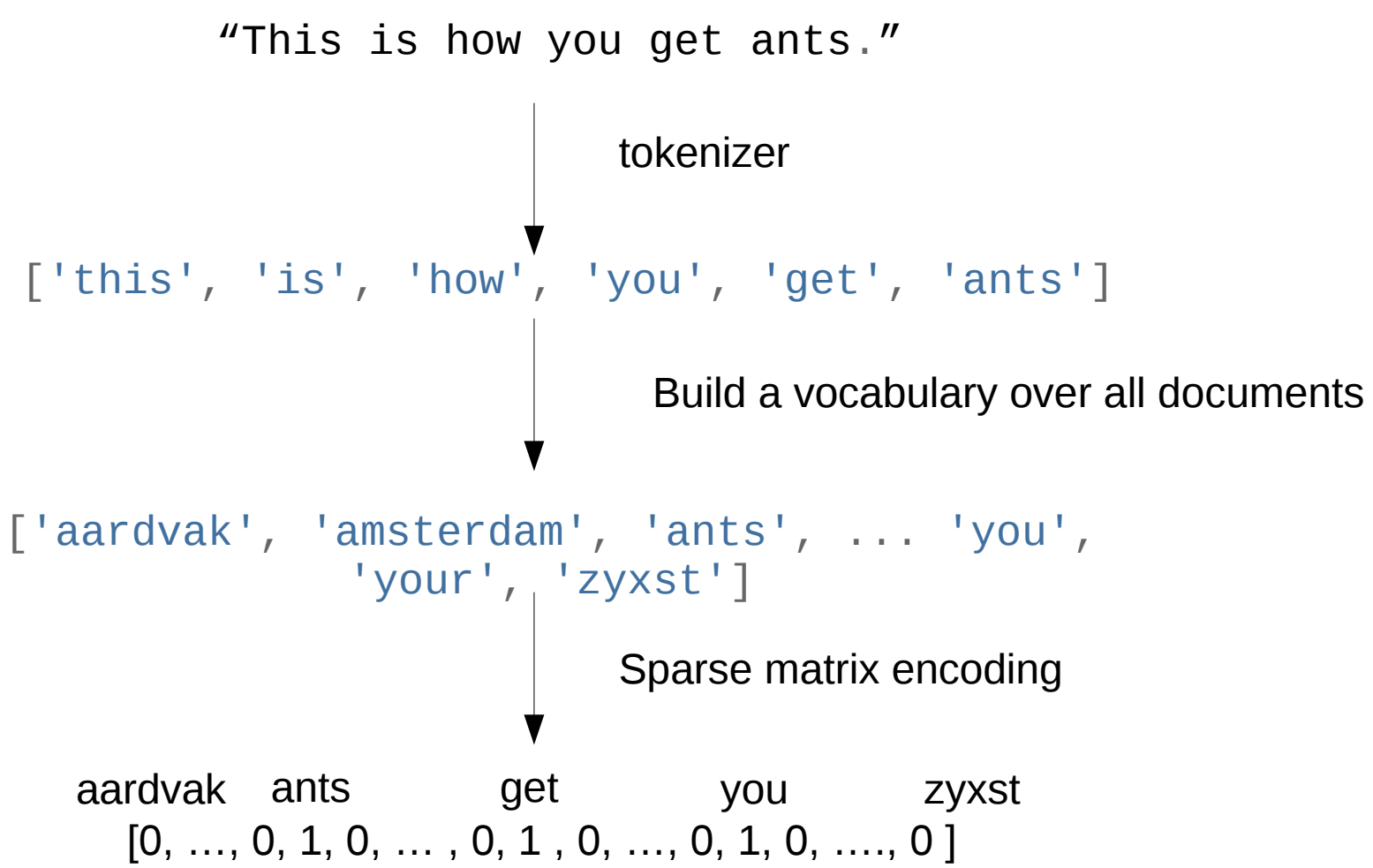
`CountVectorizer / TfidfVectorizer`

`“This is how you get ants.”`



Bag Of Word Representations

`CountVectorizer / TfidfVectorizer`



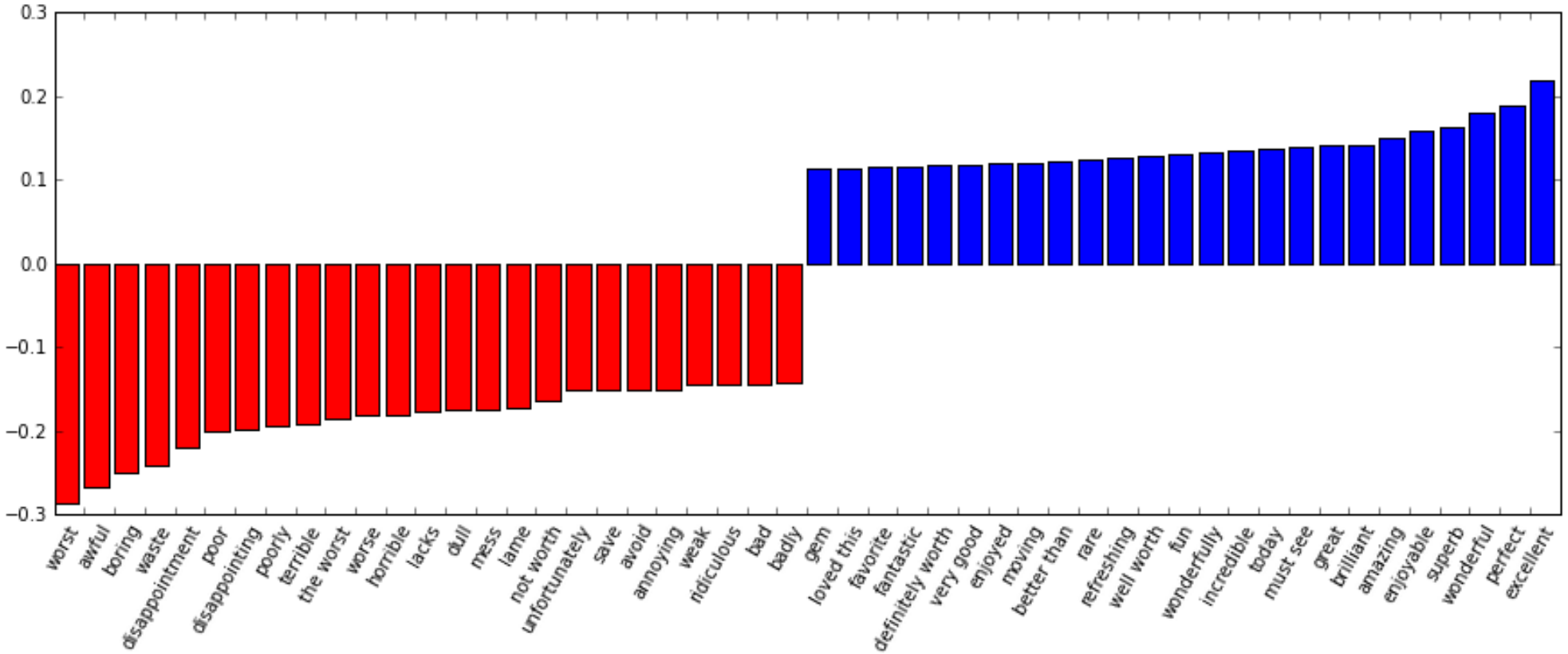
Implementation and Results

```
text_pipe = make_pipeline(CountVectorizer(), LinearSVC())
clf.fit(X_train, y_train)
clf.score(X_test, y_test)

> 0.85
```

Implementation and Results

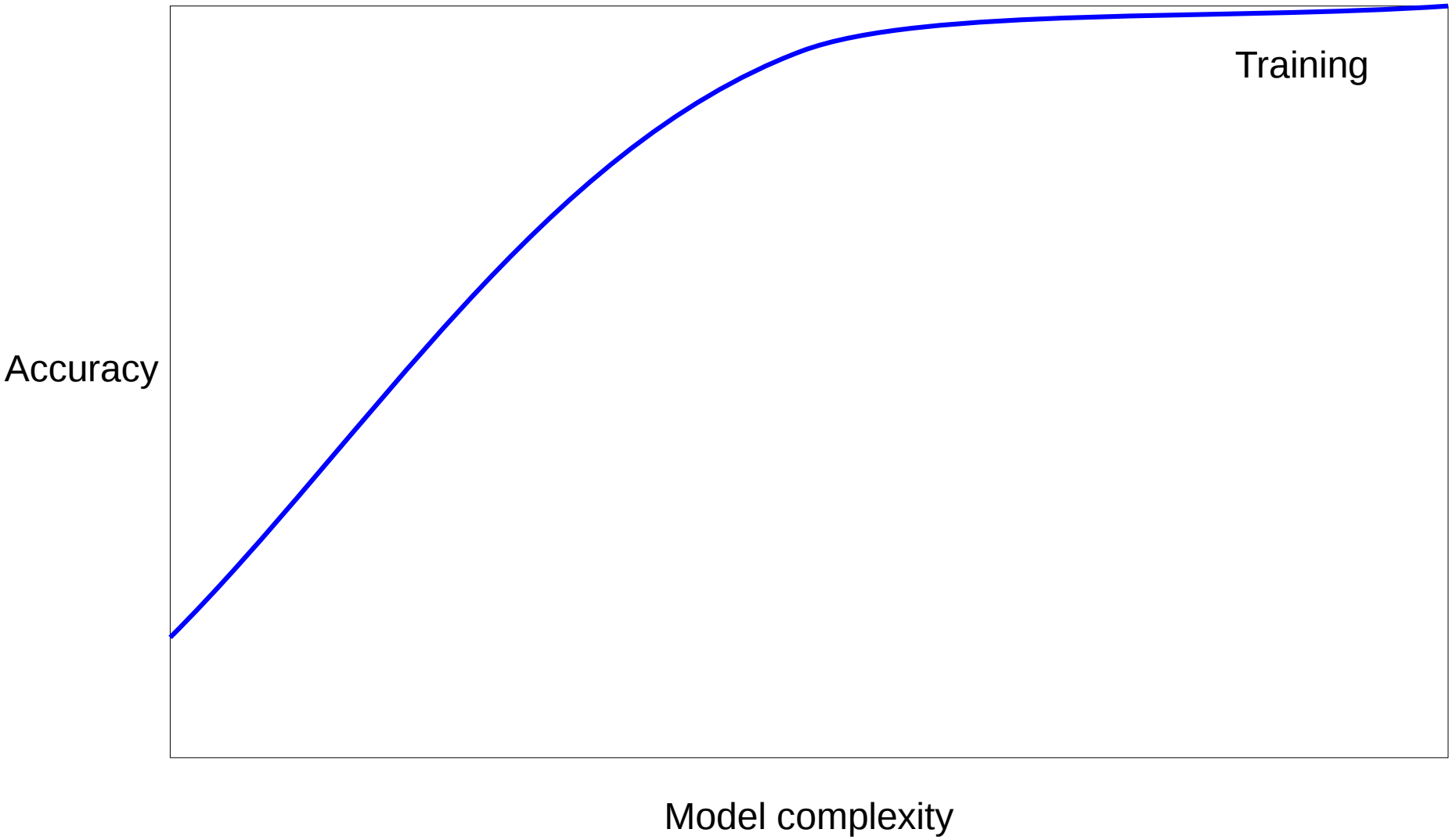
```
text_pipe = make_pipeline(CountVectorizer(), LinearSVC())
clf.fit(X_train, y_train)
clf.score(X_test, y_test)
```

```
> 0.85
```

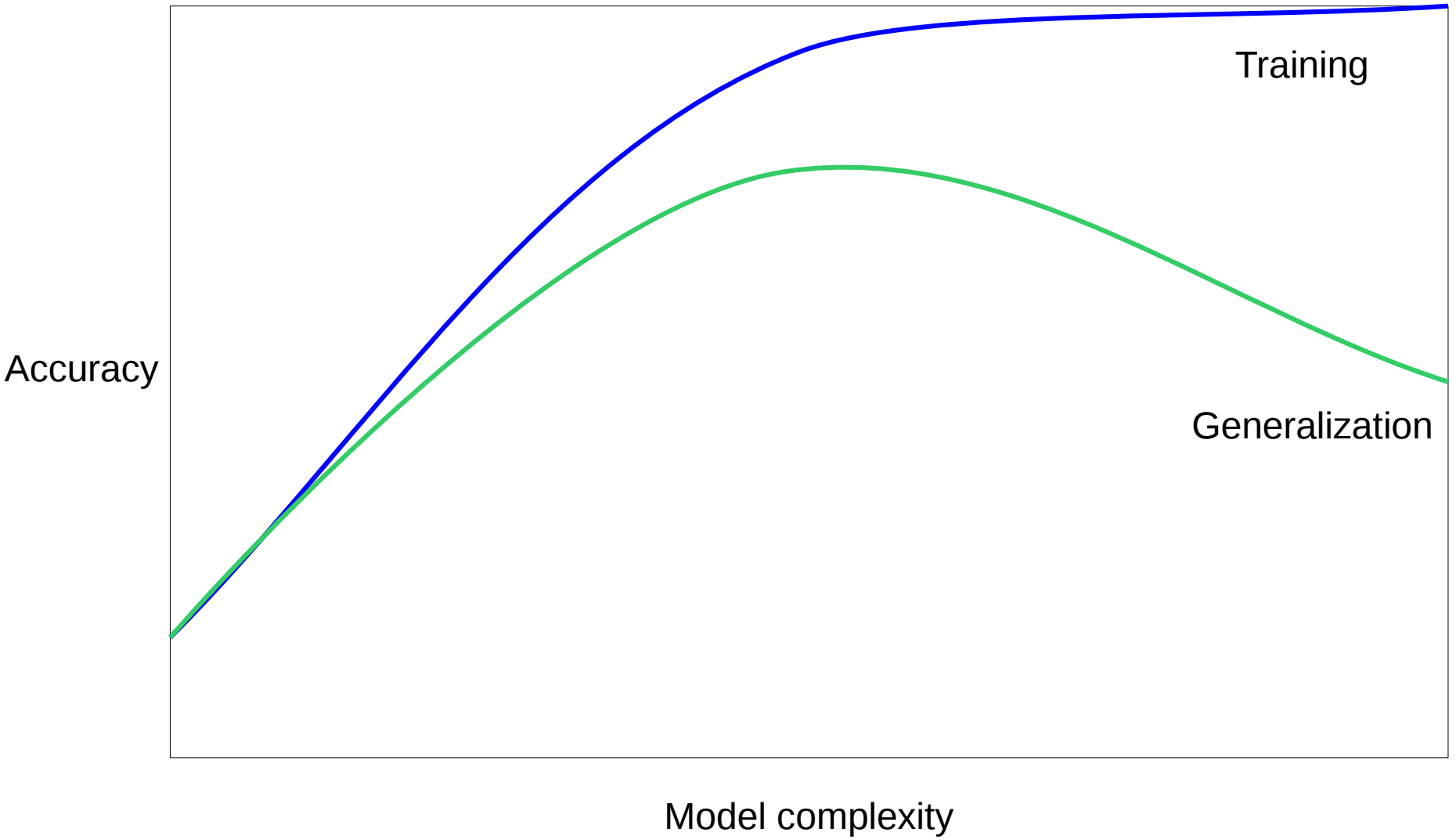


Model Complexity

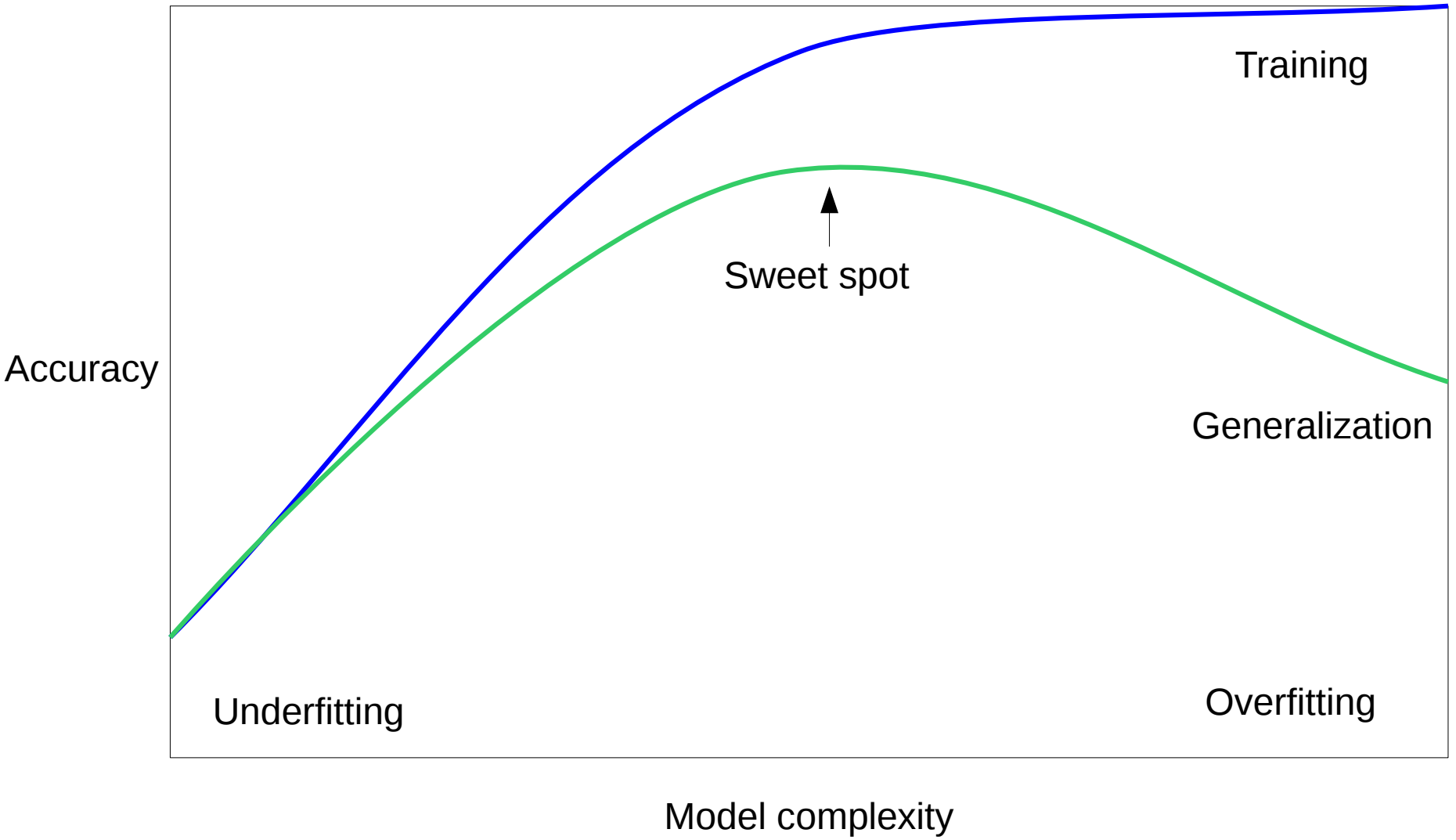
Overfitting and Underfitting



Overfitting and Underfitting



Overfitting and Underfitting



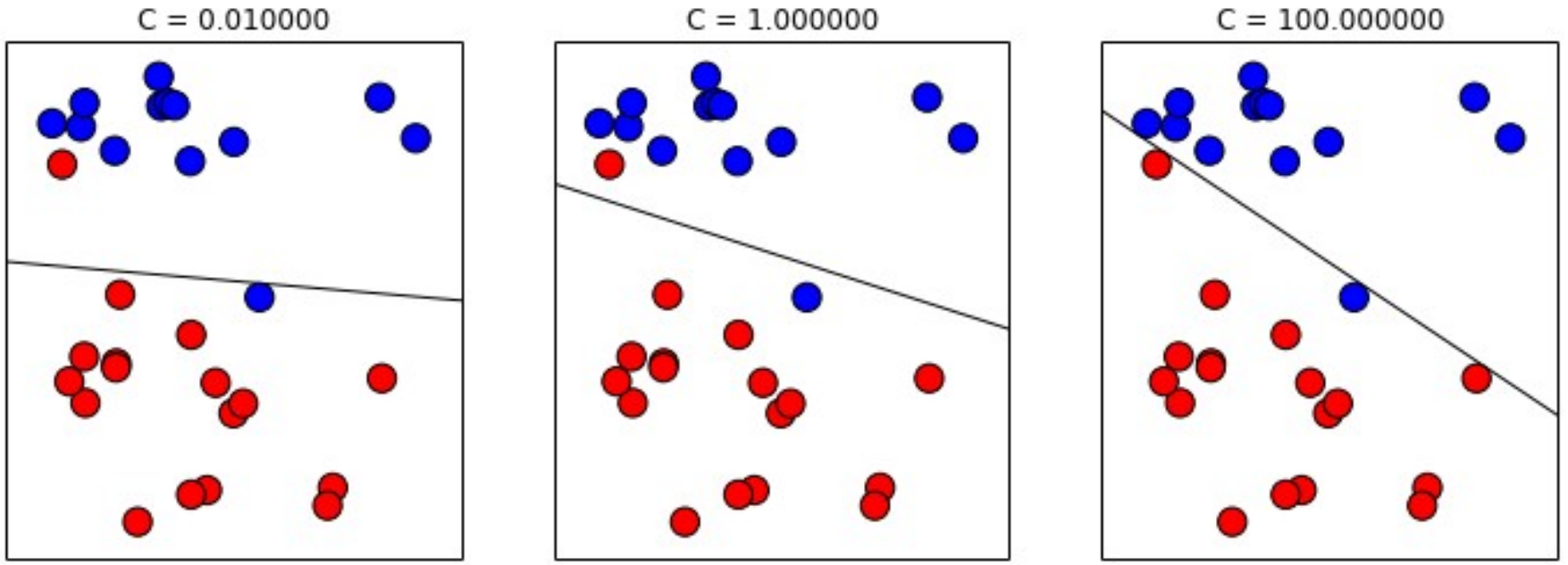
Model Complexity Examples

Linear SVM

$$\hat{y} = \text{sign}(w_0 + \sum_i w_i x_i)$$

Linear SVM

$$\hat{y} = \text{sign}(w_0 + \sum_i w_i x_i)$$



(RBF) Kernel SVM

$$\hat{y} = \text{sign}(\alpha_0 + \sum_j \alpha_j y_j k(\mathbf{x}^{(\mathbf{j})}, \mathbf{x}))$$

(RBF) Kernel SVM

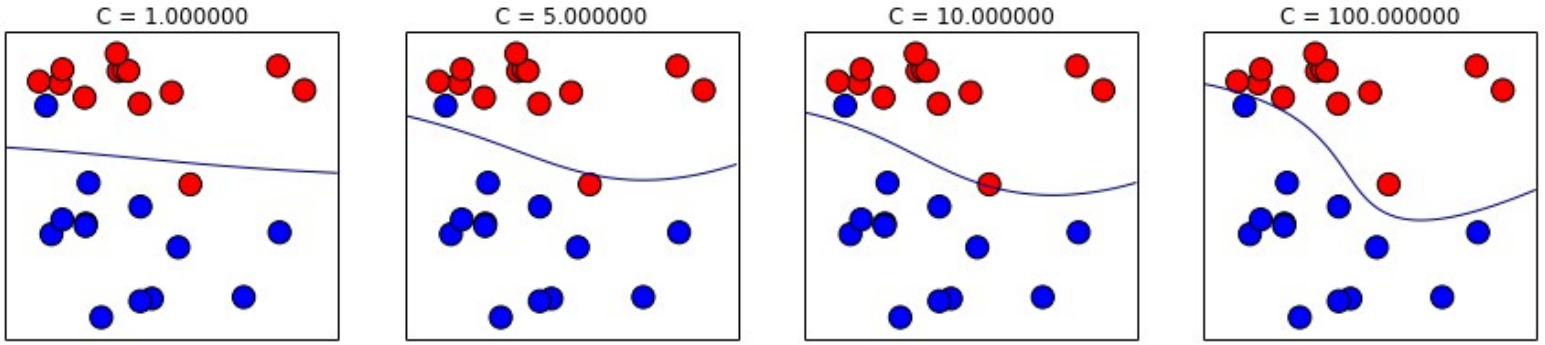
$$\hat{y} = \text{sign}(\alpha_0 + \sum_j \alpha_j y_j k(\mathbf{x^{(j)}}, \mathbf{x}))$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2)$$

(RBF) Kernel SVM

$$\hat{y} = \text{sign}(\alpha_0 + \sum_j \alpha_j y_j k(\mathbf{x}^{(\mathbf{j})}, \mathbf{x}))$$

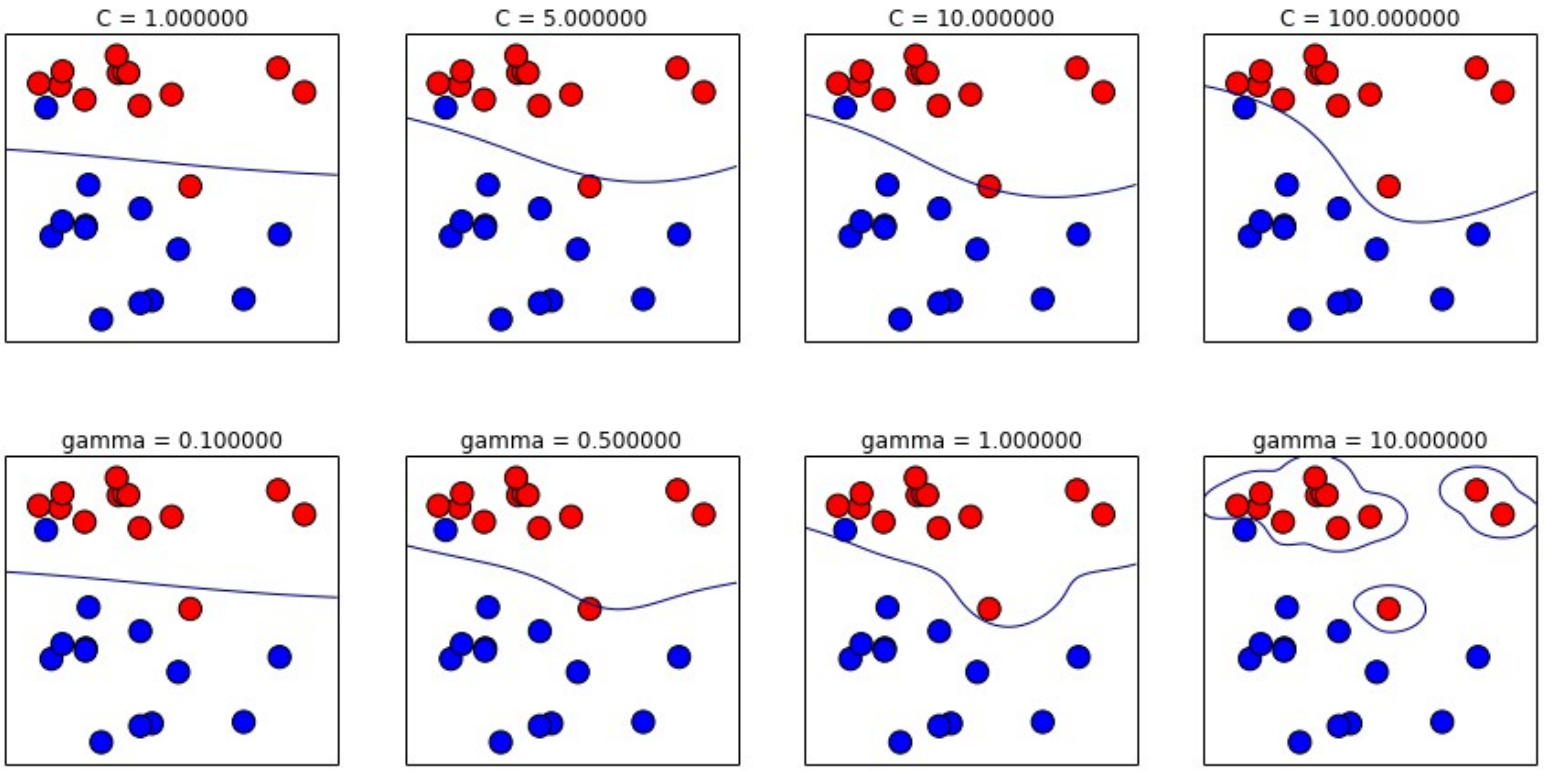
$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2)$$



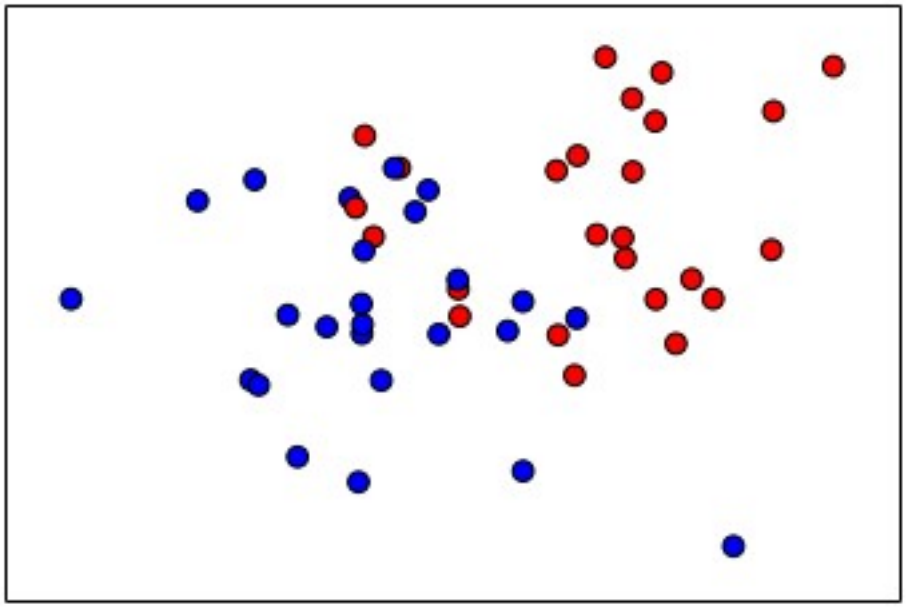
(RBF) Kernel SVM

$$\hat{y} = \text{sign}(\alpha_0 + \sum_j \alpha_j y_j k(\mathbf{x}^{(\mathbf{j})}, \mathbf{x}))$$

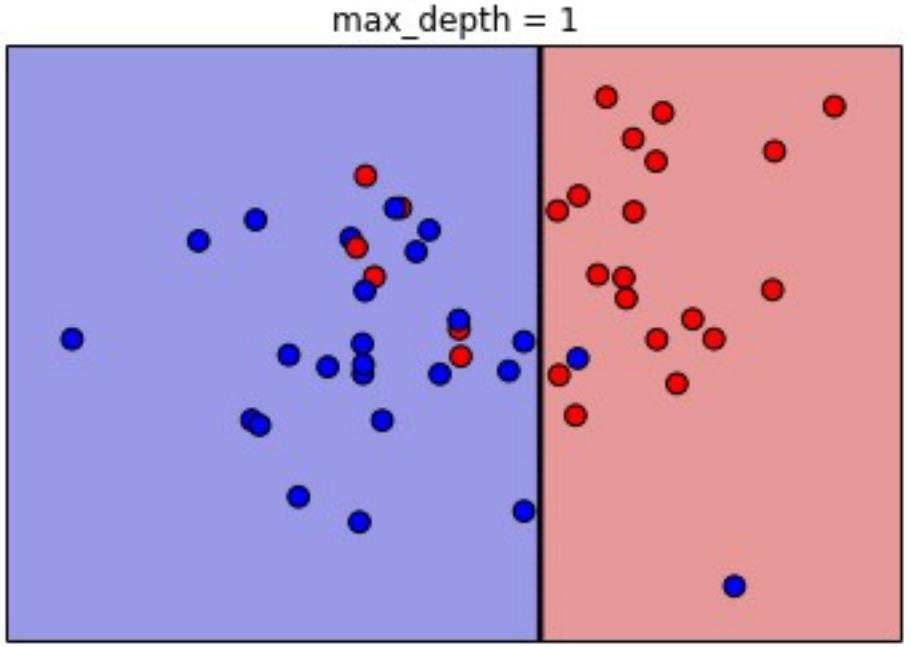
$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||^2)$$

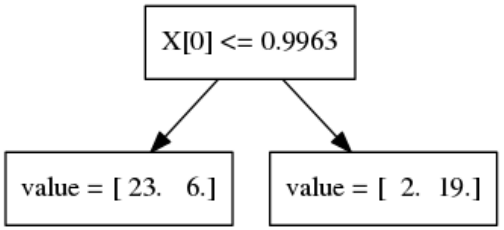


Decision Trees

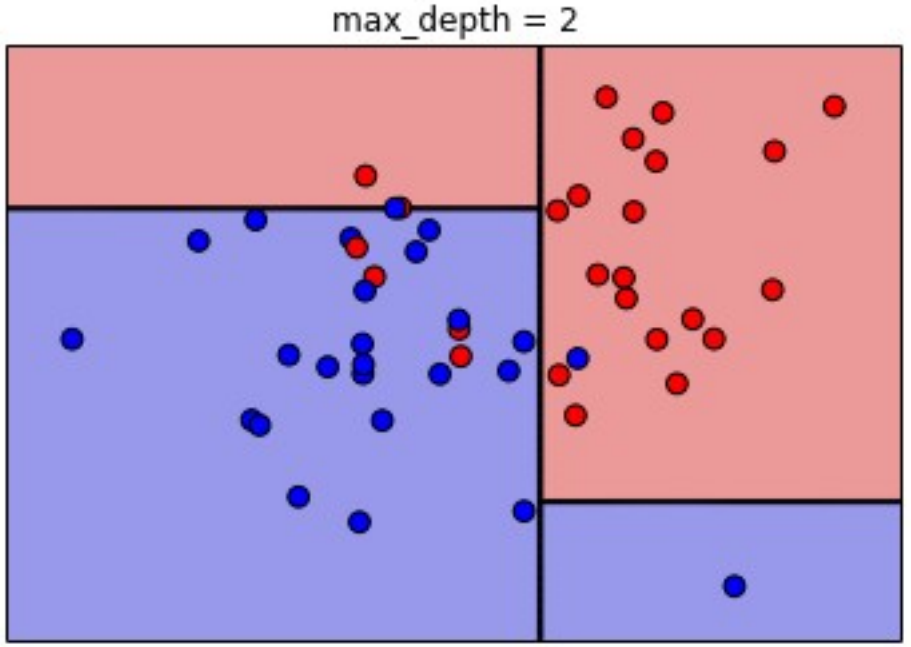


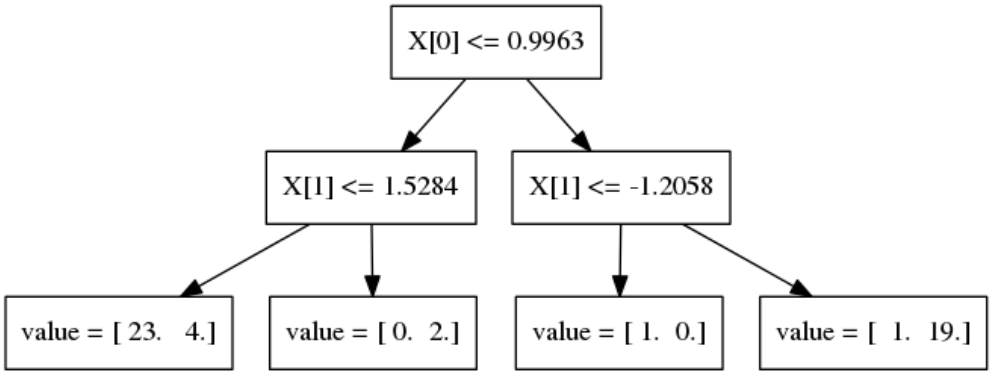
Decision Trees



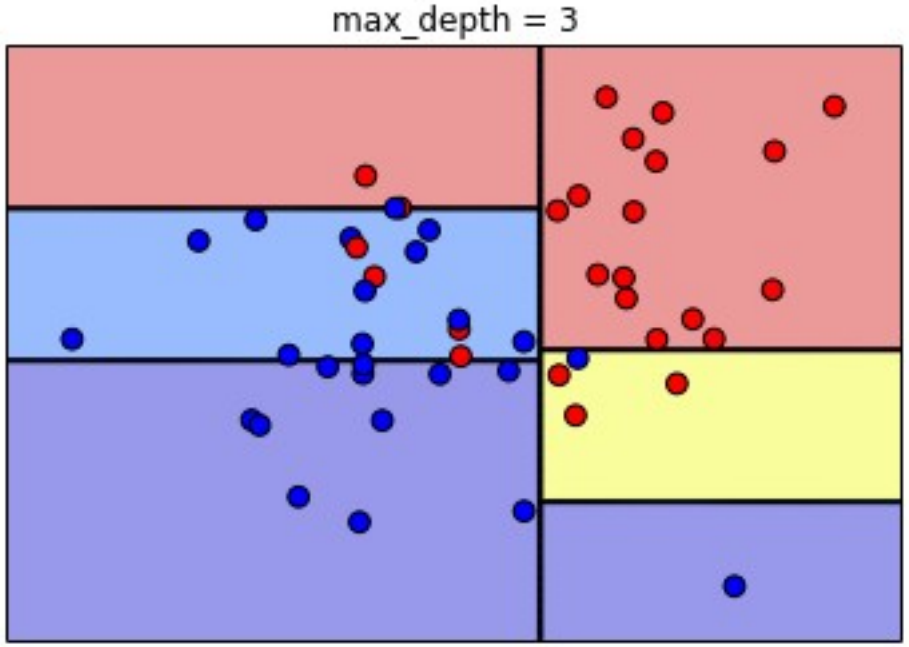


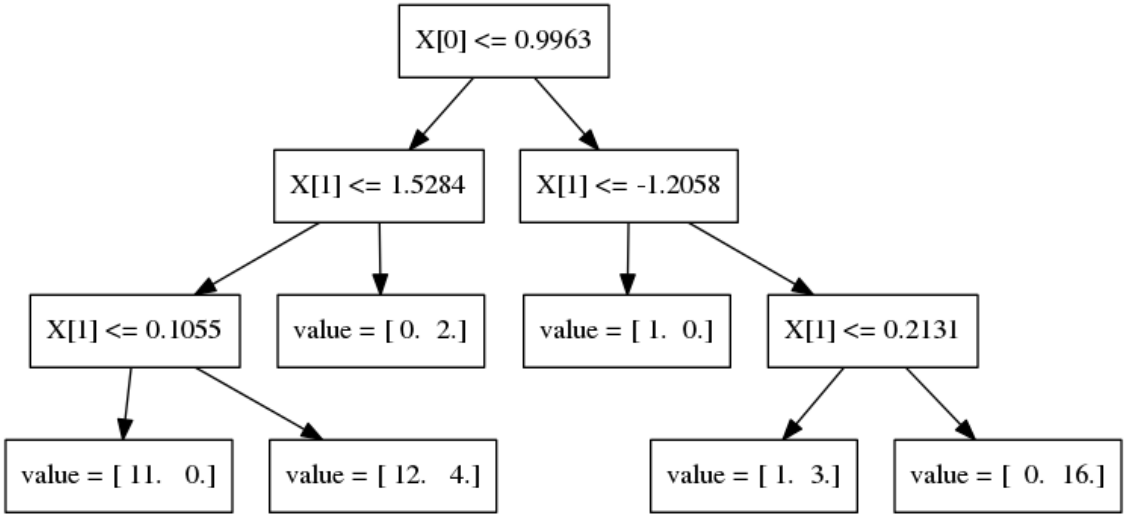
Decision Trees



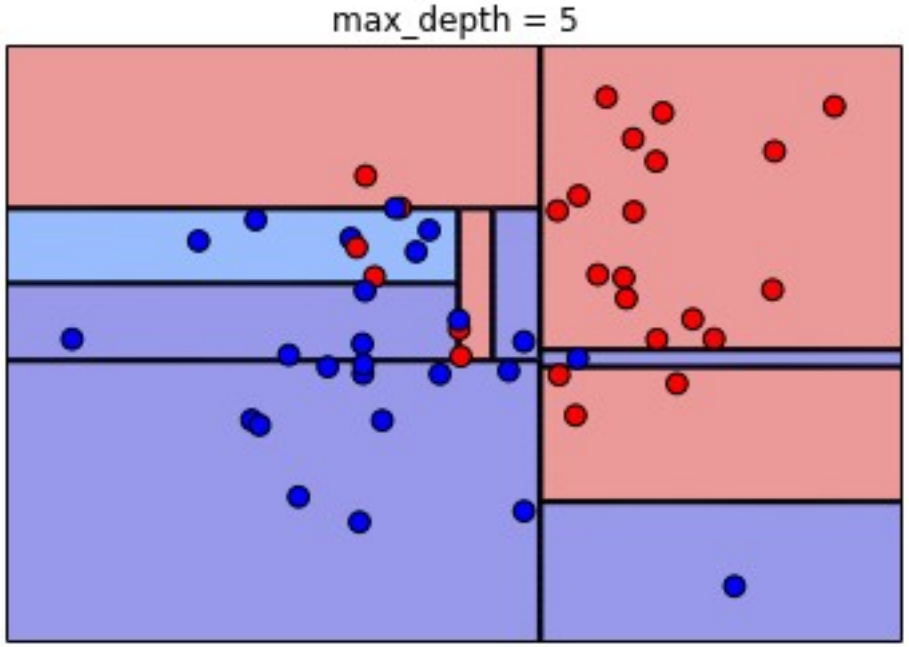


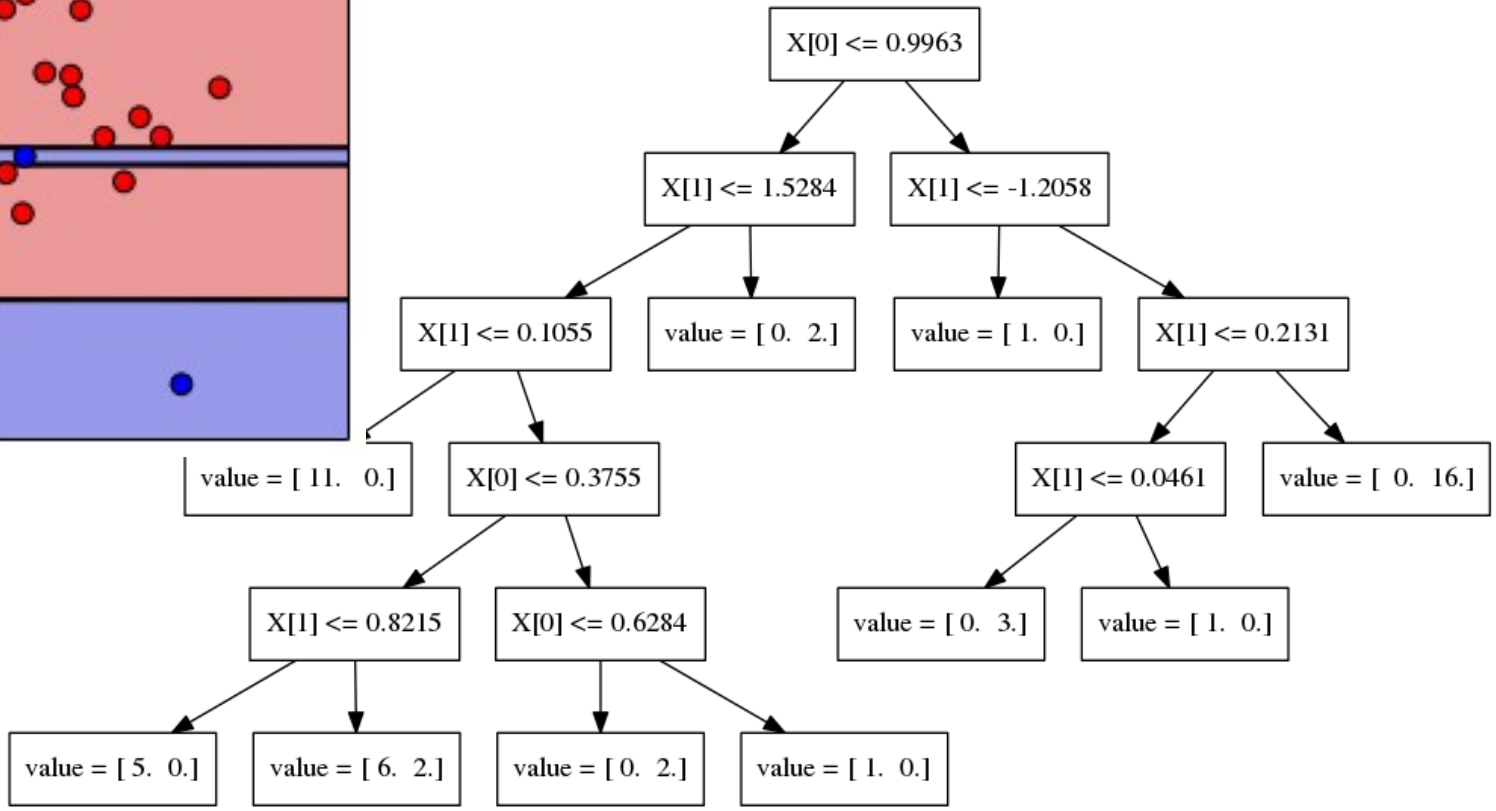
Decision Trees



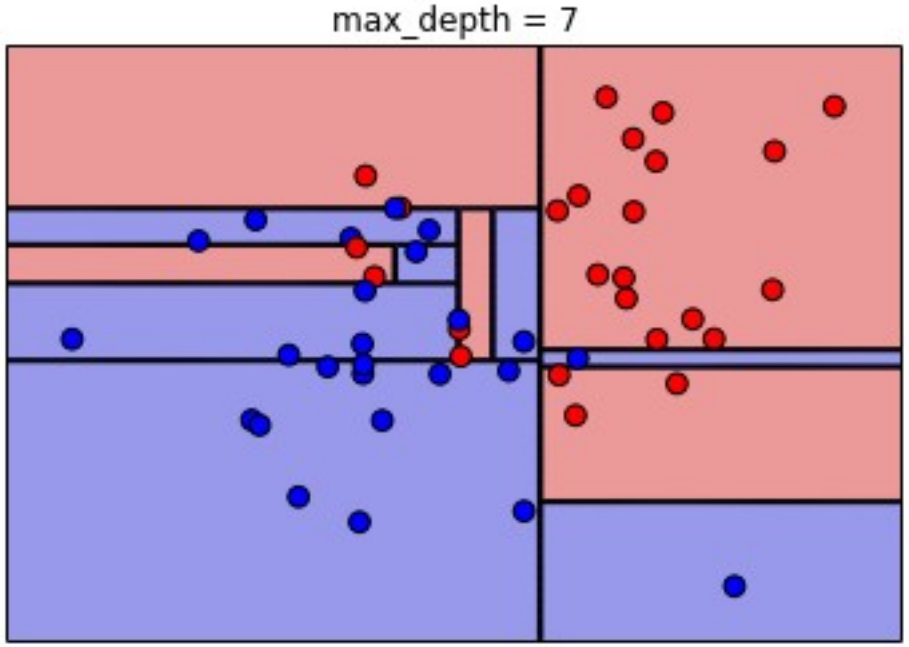


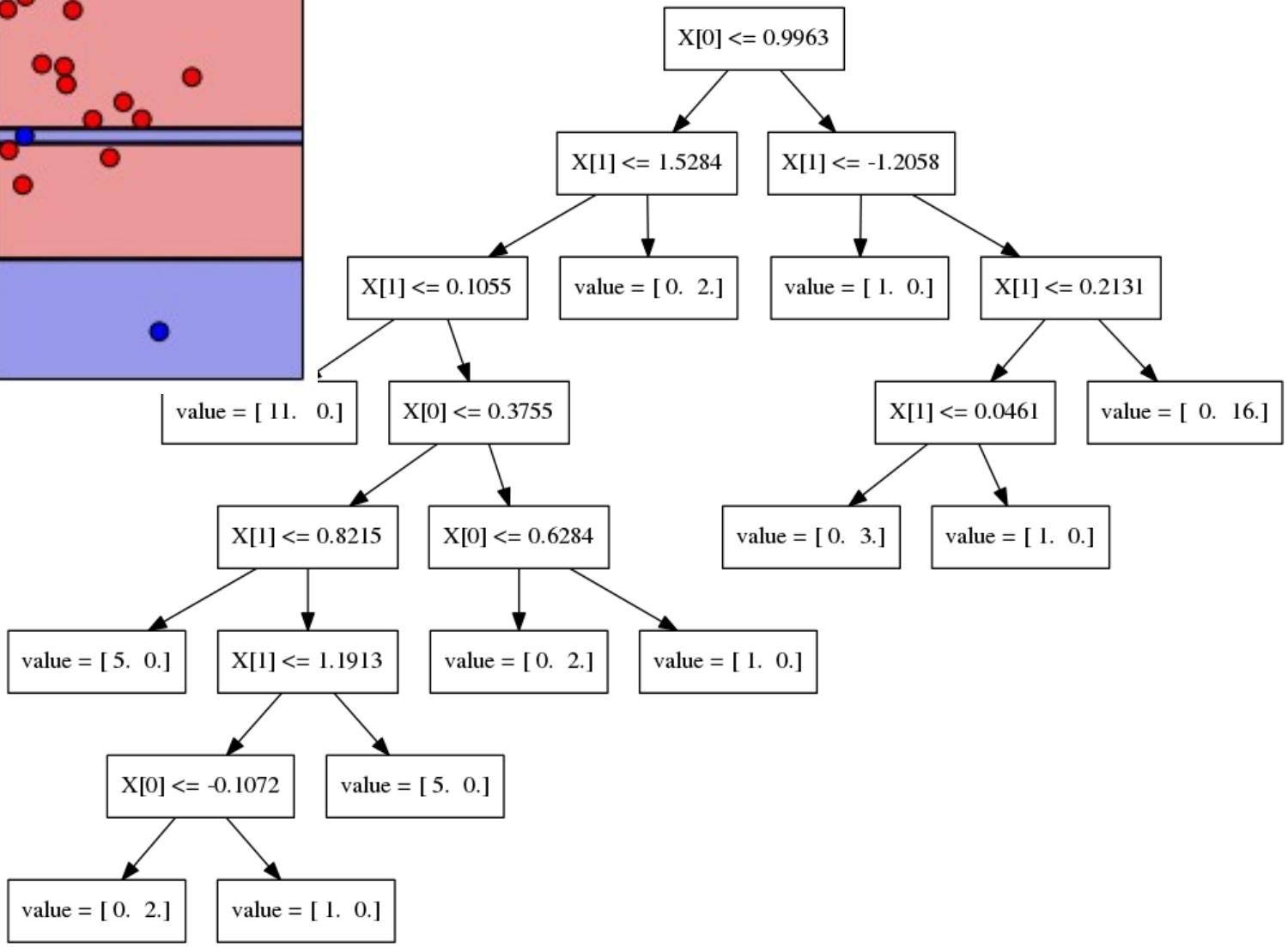
Decision Trees



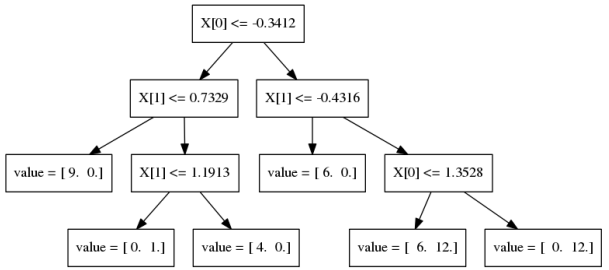


Decision Trees

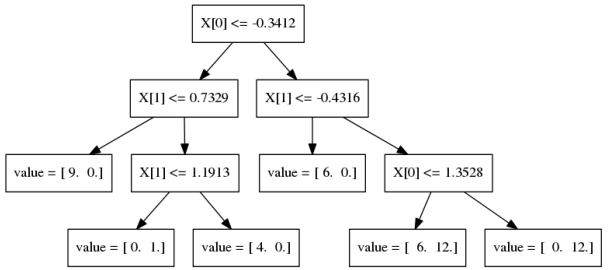


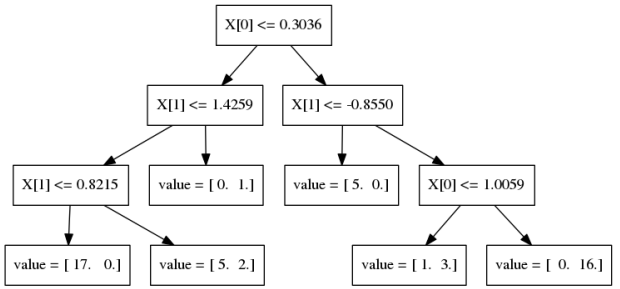


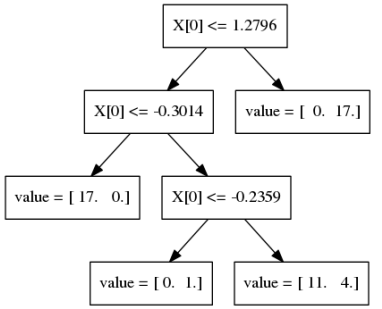
Random Forests



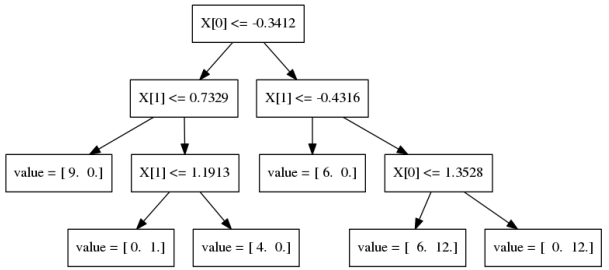
Random Forests

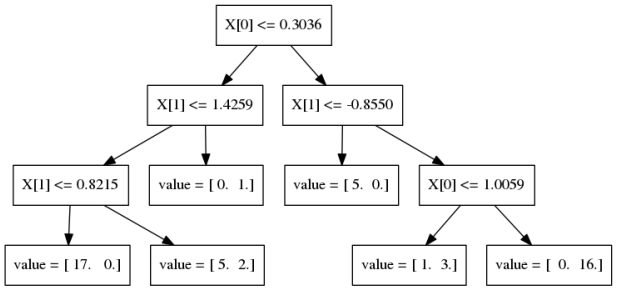


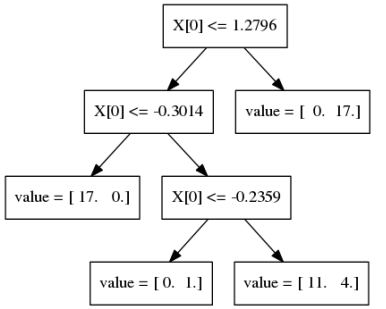


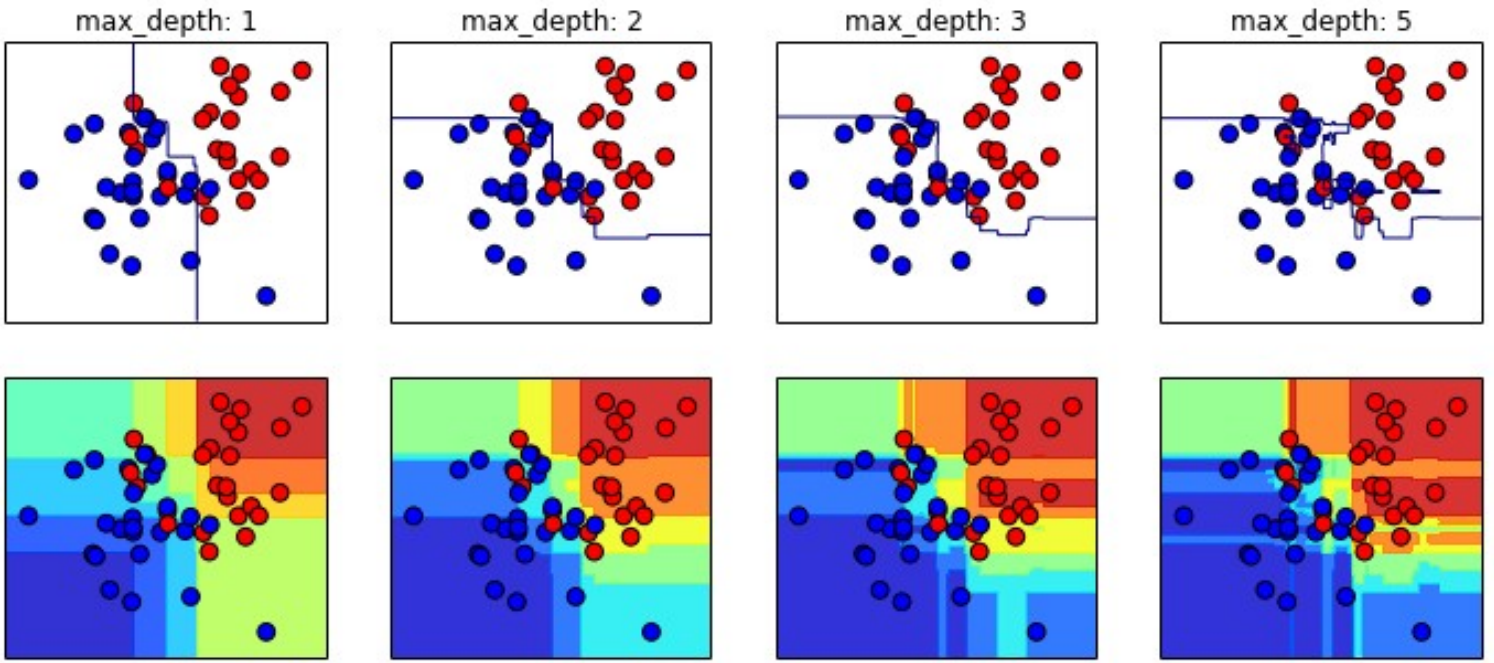


Random Forests



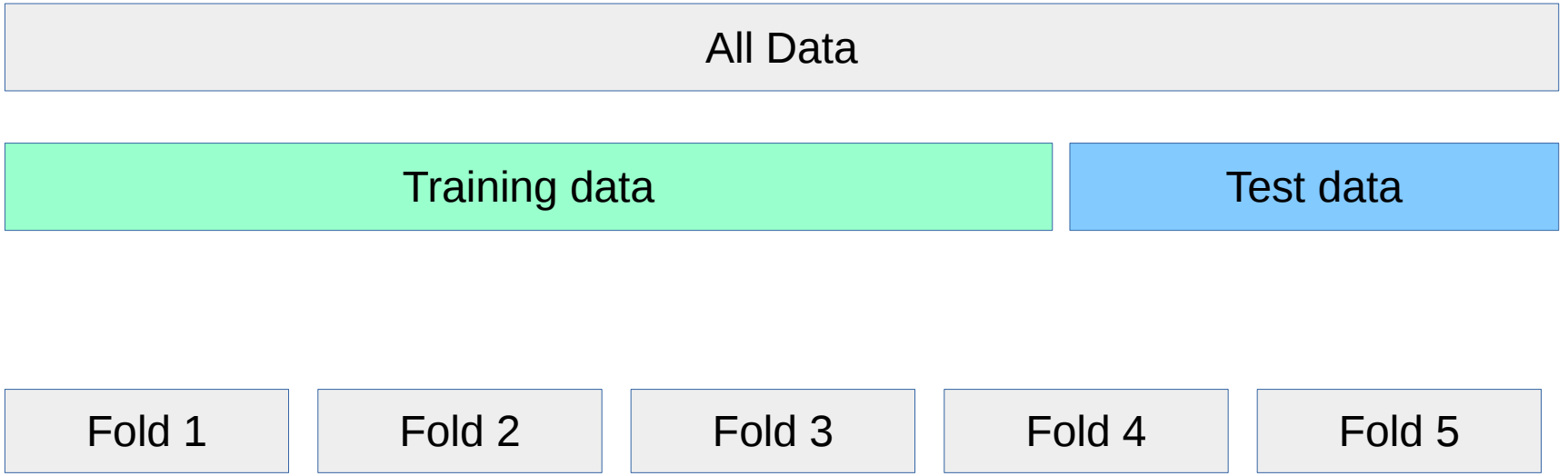


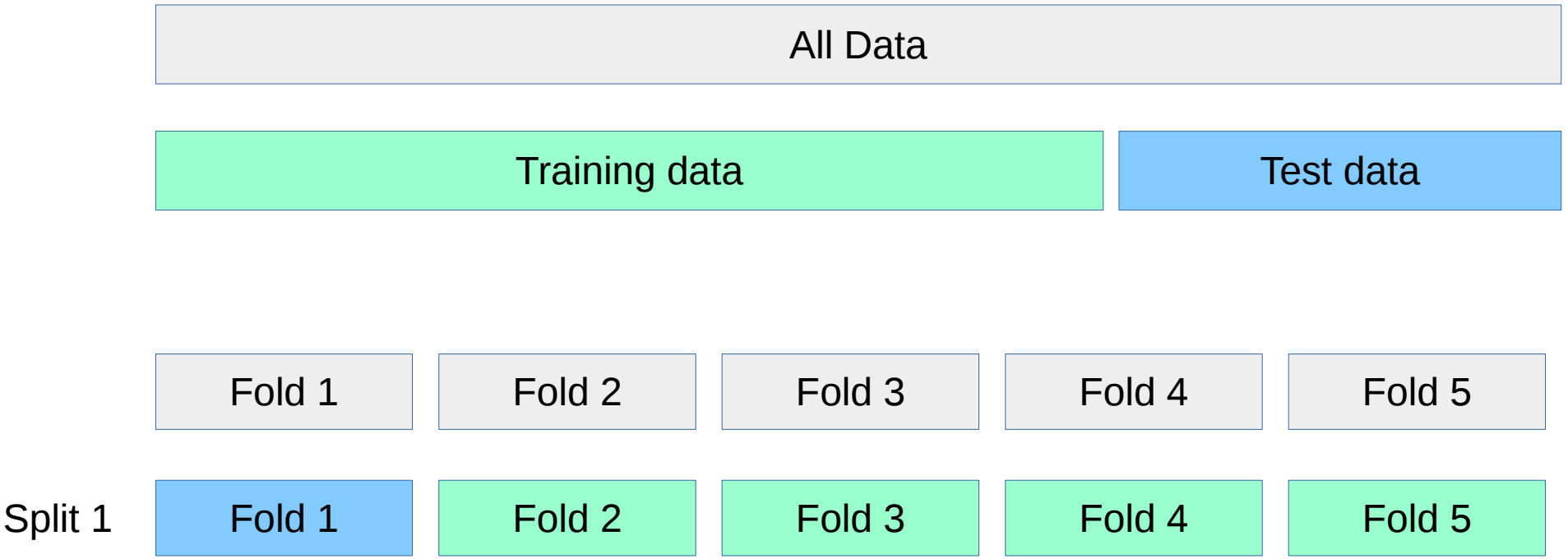


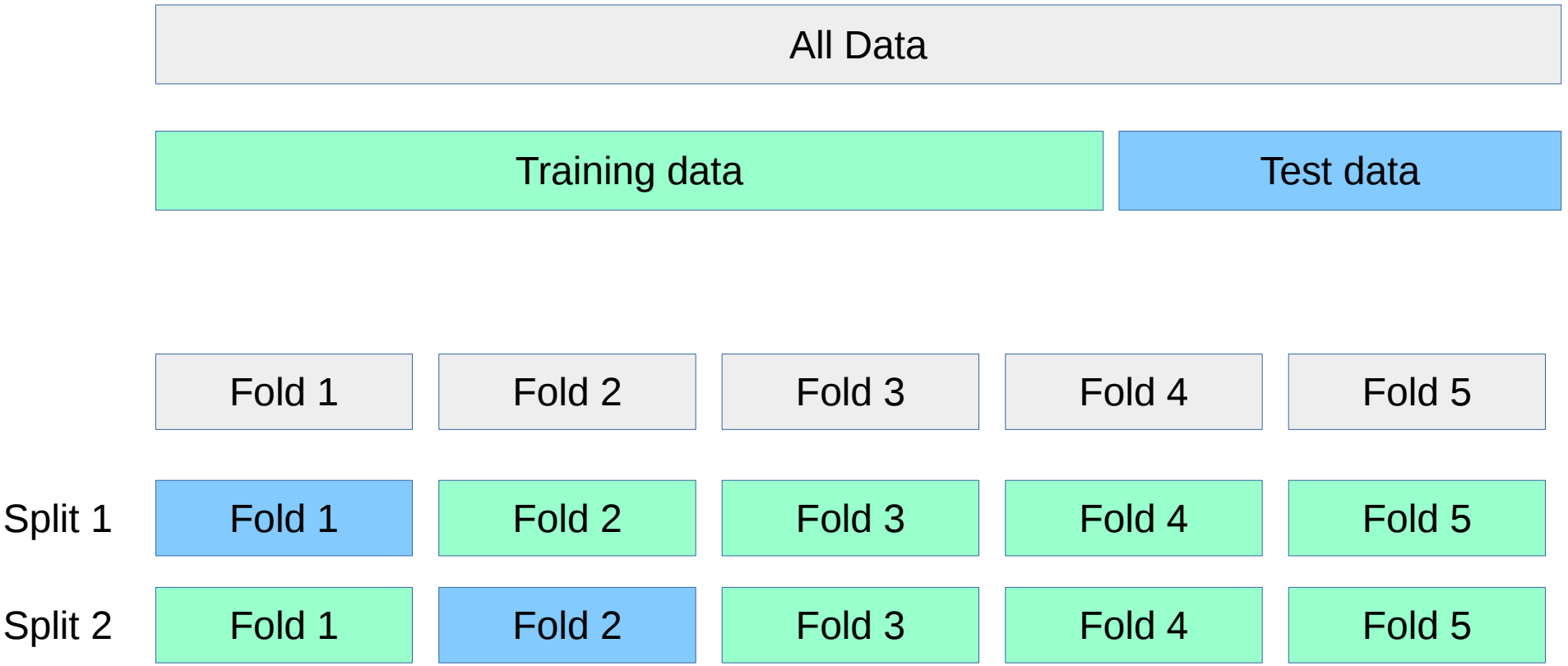


Model Evaluation and Model Selection


All Data
Training data
Test data


All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5


All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5


All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 2
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5


All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 2
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 3
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 4
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 5
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5

Cross-Validation

```
from sklearn.cross_validation import cross_val_score

scores = cross_val_score(SVC(), X, y, cv=5)
print(scores)

>> [ 0.92  1.    1.    1.    1.  ]
```

SVC(C=0.001,
gamma=0.001)

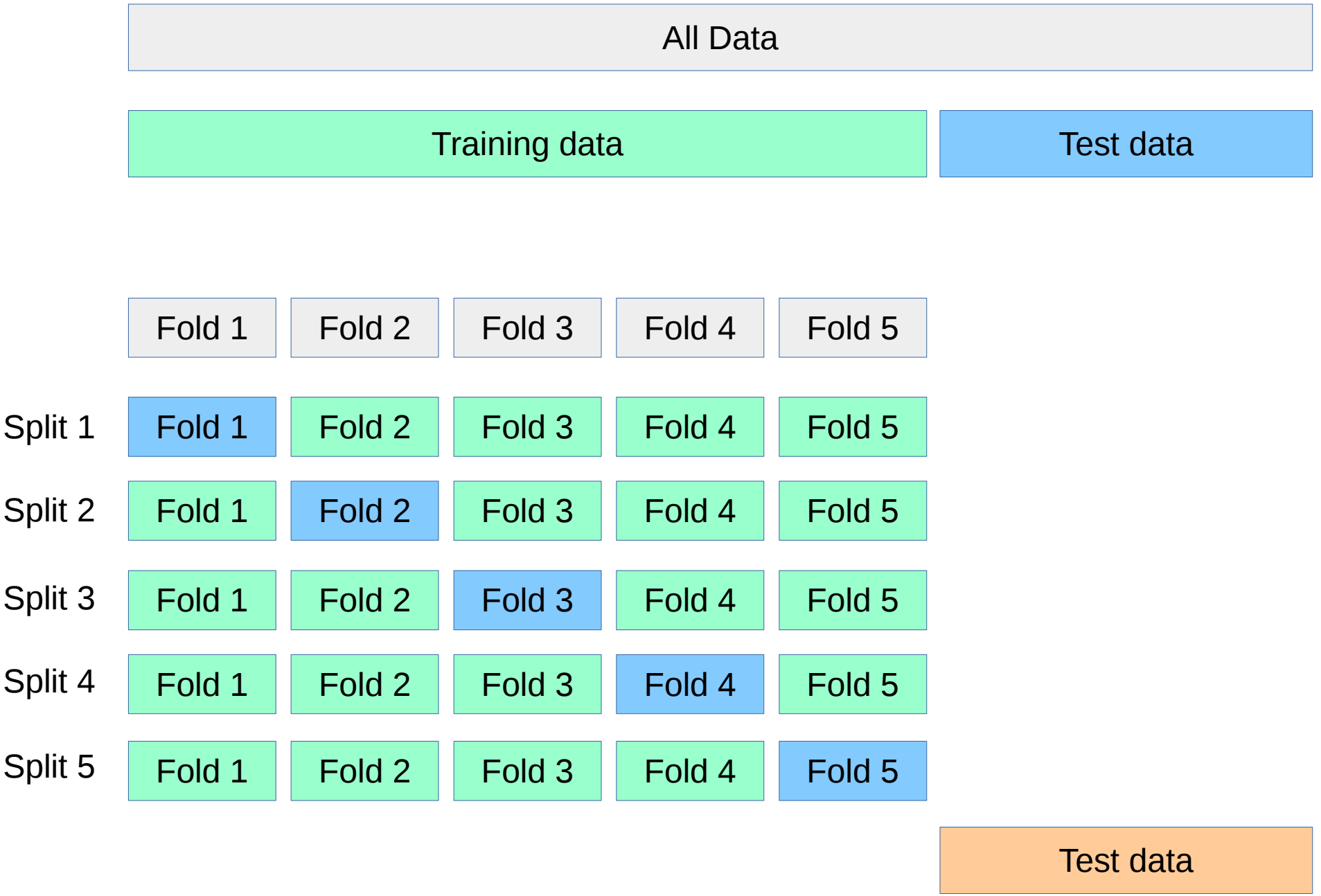
SVC(C=0.001, gamma=0.001)	SVC(C=0.01, gamma=0.001)	SVC(C=0.1, gamma=0.001)	SVC(C=1, gamma=0.001)	SVC(C=10, gamma=0.001)

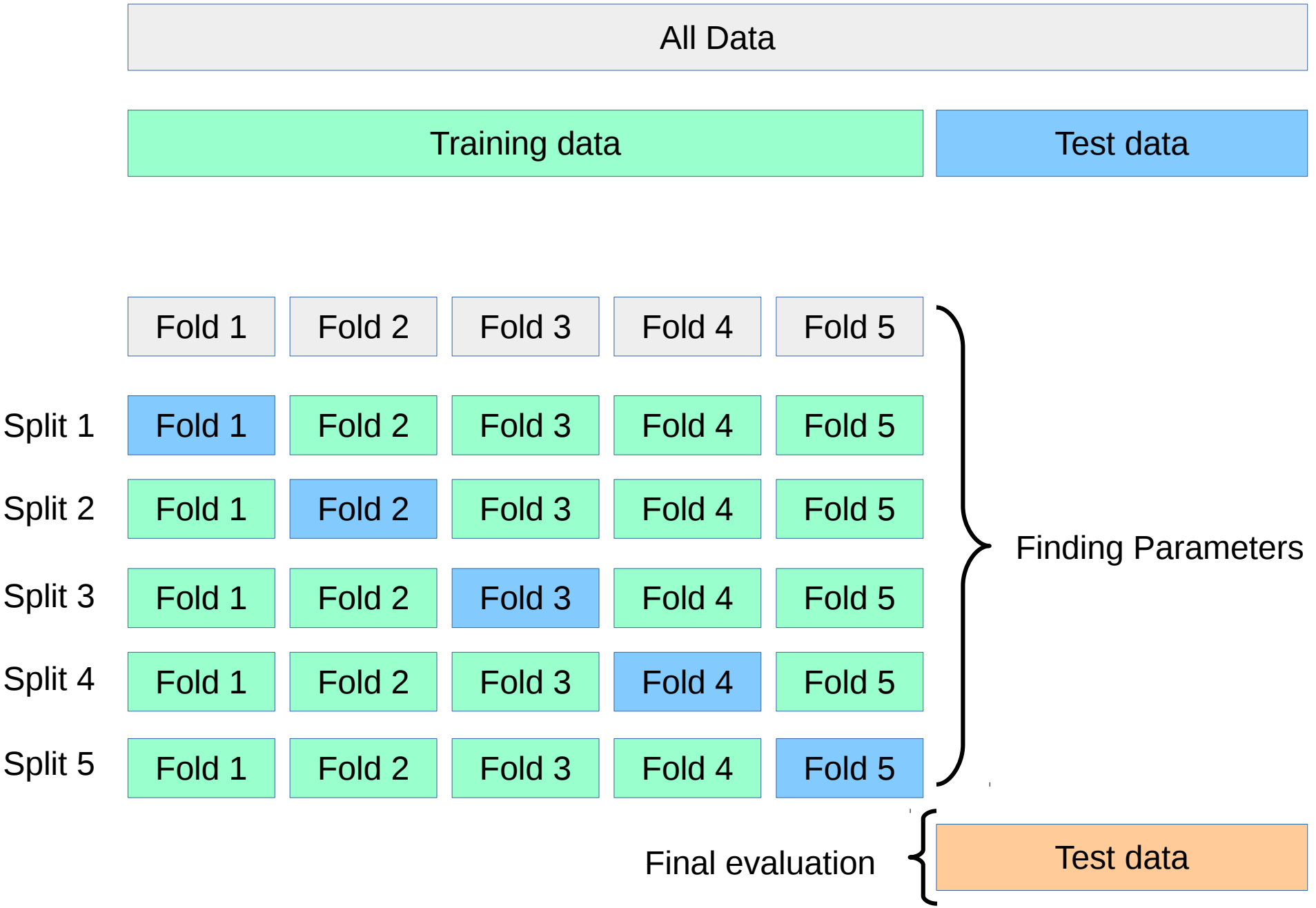
SVC(C=0.001, gamma=0.001)	SVC(C=0.01, gamma=0.001)	SVC(C=0.1, gamma=0.001)	SVC(C=1, gamma=0.001)	SVC(C=10, gamma=0.001)
SVC(C=0.001, gamma=0.01)	SVC(C=0.01, gamma=0.01)	SVC(C=0.1, gamma=0.01)	SVC(C=1, gamma=0.01)	SVC(C=10, gamma=0.01)

SVC(C=0.001, gamma=0.001)	SVC(C=0.01, gamma=0.001)	SVC(C=0.1, gamma=0.001)	SVC(C=1, gamma=0.001)	SVC(C=10, gamma=0.001)
SVC(C=0.001, gamma=0.01)	SVC(C=0.01, gamma=0.01)	SVC(C=0.1, gamma=0.01)	SVC(C=1, gamma=0.01)	SVC(C=10, gamma=0.01)
SVC(C=0.001, gamma=0.1)	SVC(C=0.01, gamma=0.1)	SVC(C=0.1, gamma=0.1)	SVC(C=1, gamma=0.1)	SVC(C=10, gamma=0.1)

SVC(C=0.001, gamma=0.001)	SVC(C=0.01, gamma=0.001)	SVC(C=0.1, gamma=0.001)	SVC(C=1, gamma=0.001)	SVC(C=10, gamma=0.001)
SVC(C=0.001, gamma=0.01)	SVC(C=0.01, gamma=0.01)	SVC(C=0.1, gamma=0.01)	SVC(C=1, gamma=0.01)	SVC(C=10, gamma=0.01)
SVC(C=0.001, gamma=0.1)	SVC(C=0.01, gamma=0.1)	SVC(C=0.1, gamma=0.1)	SVC(C=1, gamma=0.1)	SVC(C=10, gamma=0.1)
SVC(C=0.001, gamma=1)	SVC(C=0.01, gamma=1)	SVC(C=0.1, gamma=1)	SVC(C=1, gamma=1)	SVC(C=10, gamma=1)
SVC(C=0.001, gamma=10)	SVC(C=0.01, gamma=10)	SVC(C=0.1, gamma=10)	SVC(C=1, gamma=10)	SVC(C=10, gamma=10)


All Data
Training data
Test data

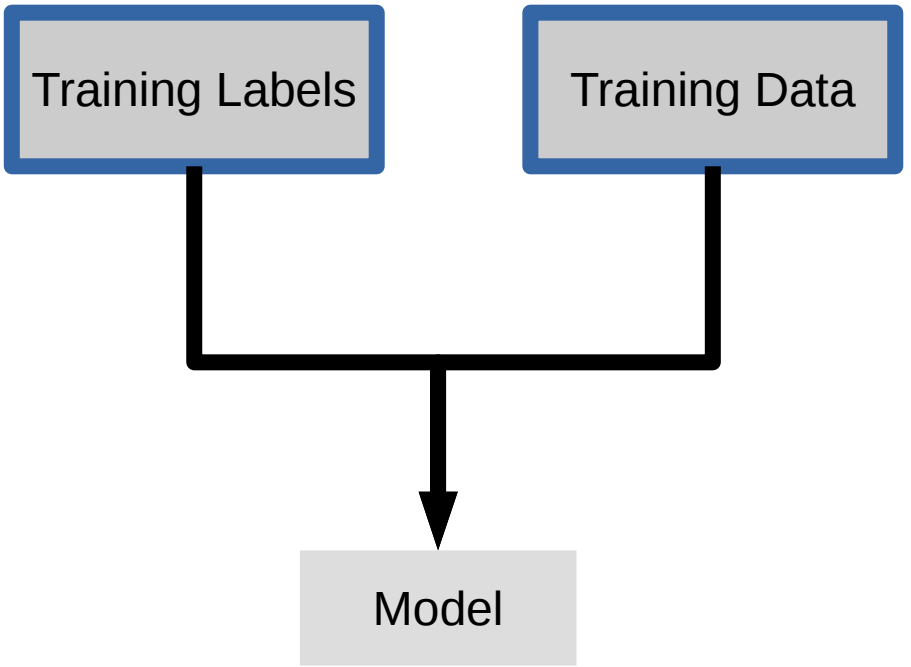

All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5
Test data

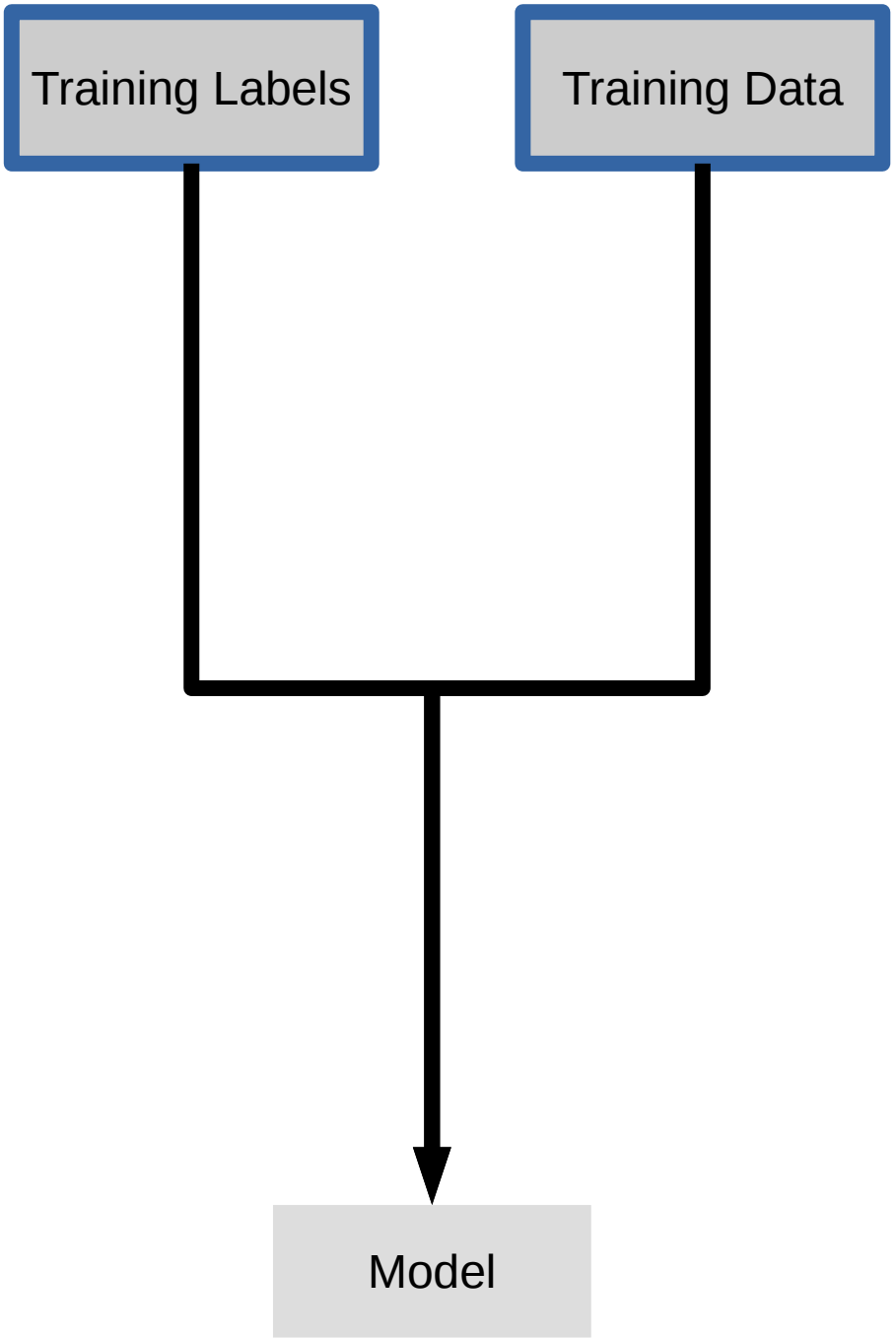

All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5
Finding Parameters
Final evaluation
Test data

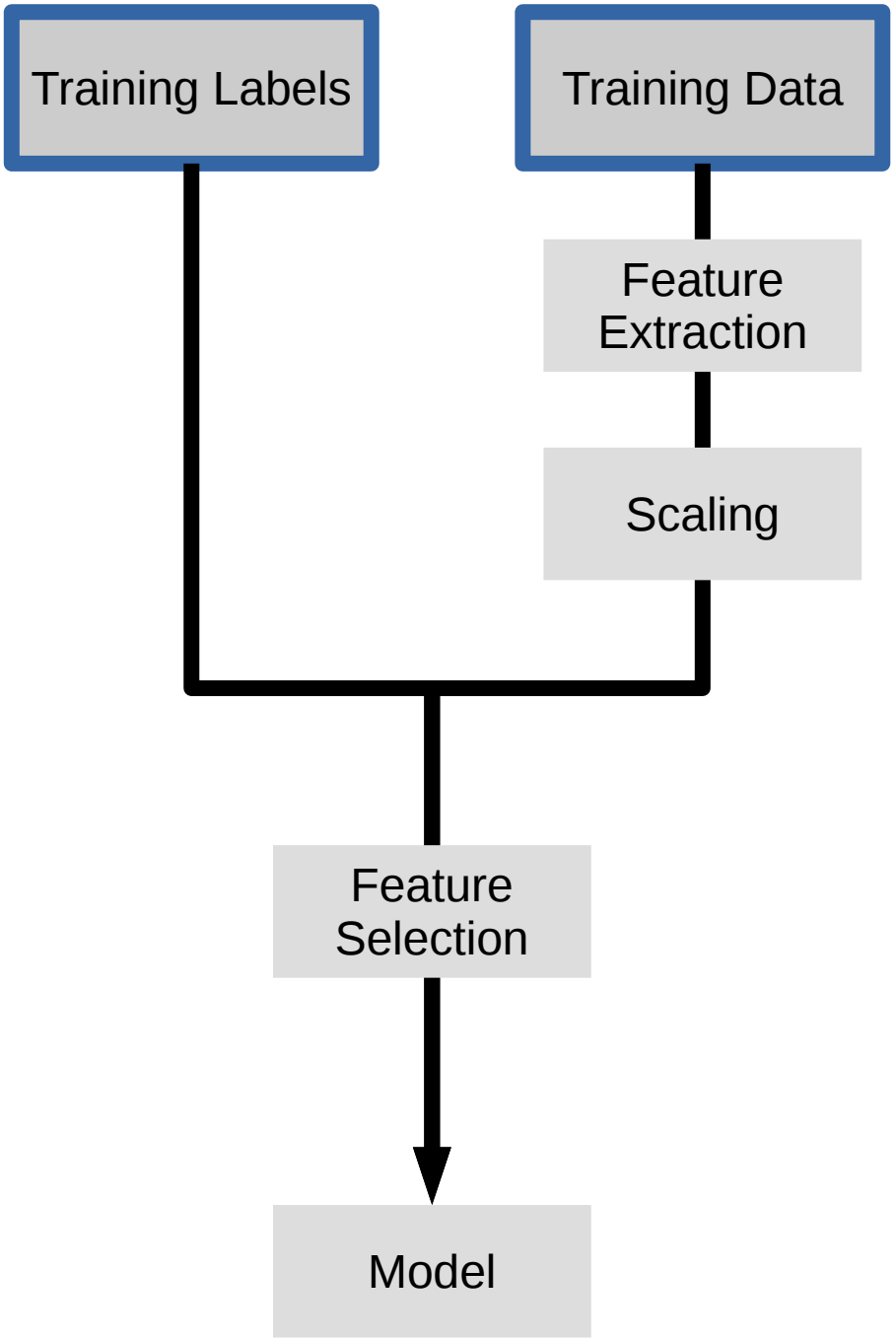
Cross -Validated Grid Search

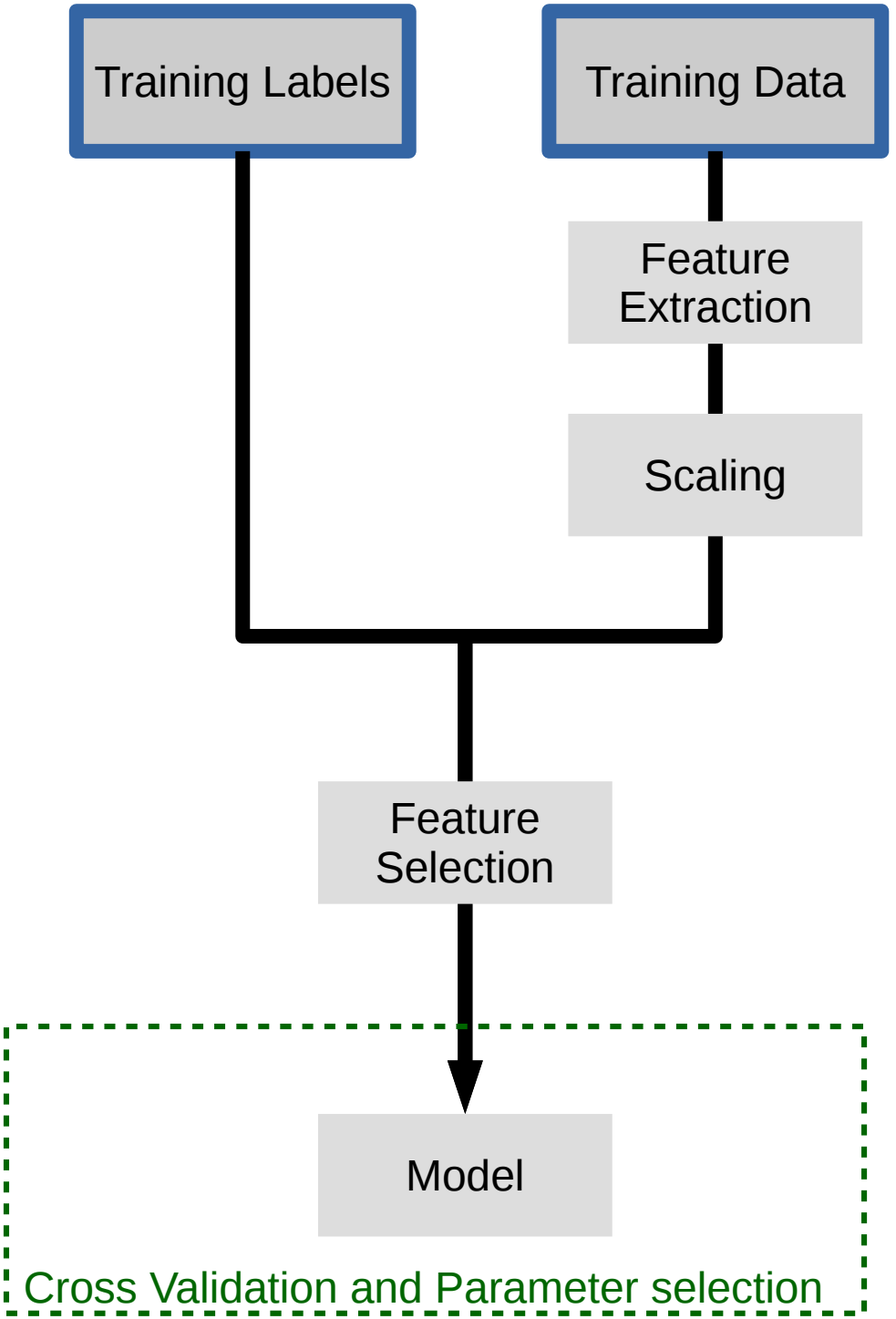
```
from sklearn.grid_search import GridSearchCV
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)

param_grid = {'C': 10. ** np.arange(-3, 3),
              'gamma': 10. ** np.arange(-3, 3)}
grid = GridSearchCV(SVC(), param_grid=param_grid)
grid.fit(X_train, y_train)
grid.score(X_test, y_test)
```

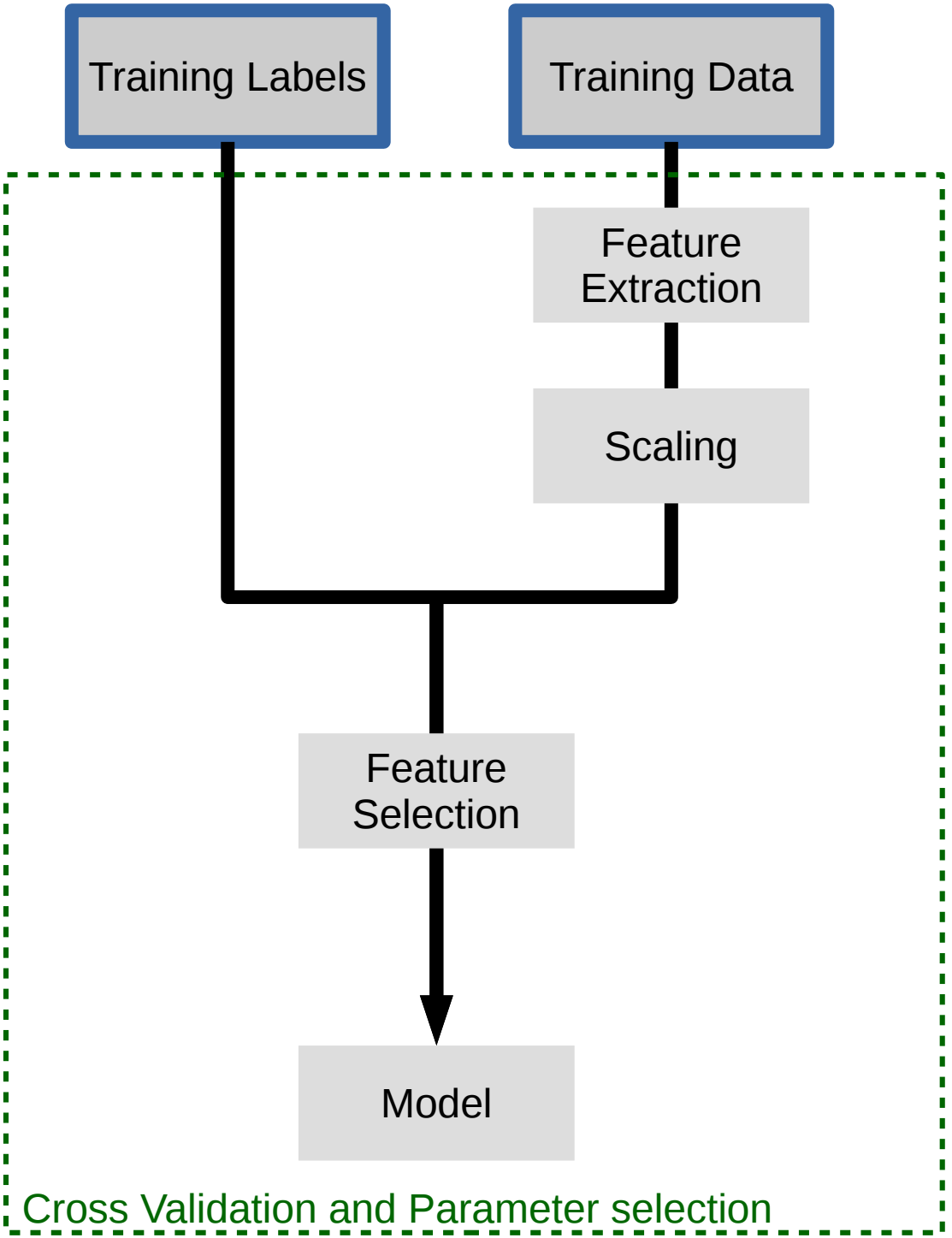
Pipelines


Training Labels
Training Data
Model


Training Labels
Training Data
Model


Training Labels
Training Data
Feature Extraction
Scaling
Feature Selection
Model


Training Labels
Training Data
Feature Extraction
Scaling
Feature Selection
Model
Cross Validation and Parameter selection

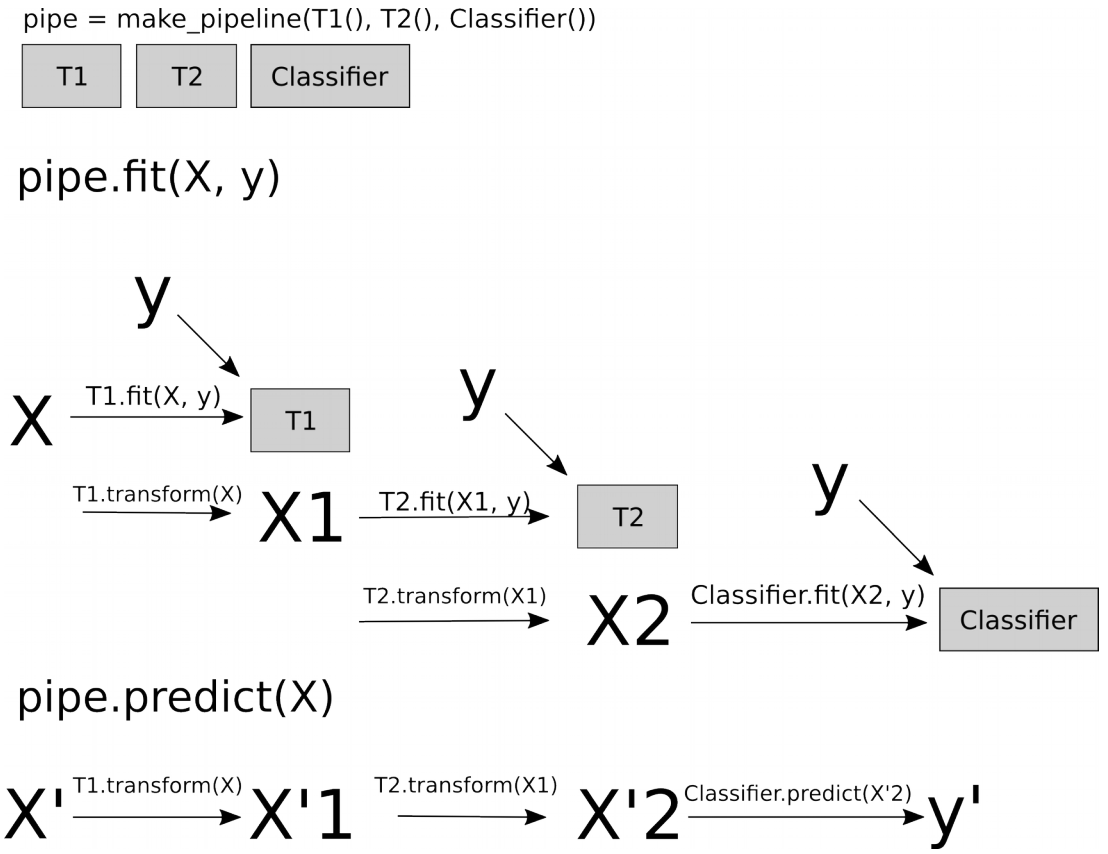

Training Labels
Training Data
Feature Extraction
Scaling
Feature Selection
Model
Cross Validation and Parameter selection

Pipelines

```
from sklearn.pipeline import make_pipeline

pipe = make_pipeline(StandardScaler(), SVC())
pipe.fit(X_train, y_train)
pipe.predict(X_test)
```

Pipelines

```
from sklearn.pipeline import make_pipeline

pipe = make_pipeline(StandardScaler(), SVC())
pipe.fit(X_train, y_train)
pipe.predict(X_test)
```



Combining Pipelines and Grid Search

Proper cross-validation

```
param_grid = {'svc__C': 10. ** np.arange(-3, 3),
              'svc__gamma': 10. ** np.arange(-3, 3)}

scaler_pipe = make_pipeline(StandardScaler(), SVC())
grid = GridSearchCV(scaler_pipe, param_grid=param_grid, cv=5)
grid.fit(X_train, y_train)
```

Combining Pipelines and Grid Search II

Searching over parameters of the preprocessing step

```
param_grid = {'selectkbest__k': [1, 2, 3, 4],
              'svc__C': 10. ** np.arange(-3, 3),
              'svc__gamma': 10. ** np.arange(-3, 3)}

scaler_pipe = make_pipeline(SelectKBest(), SVC())
grid = GridSearchCV(scaler_pipe, param_grid=param_grid, cv=5)
grid.fit(X_train, y_train)
```

Do cross-validation over all steps jointly.
Keep a separate test set until the very end.

Scoring Functions

GridSeachCV
cross_val_score

Default:
Accuracy (classification)
R2 (regression)

Scoring with imbalanced data

```
cross_val_score(SVC(), X_train, y_train)
>>> array([ 0.9,  0.9,  0.9])
```

Scoring with imbalanced data

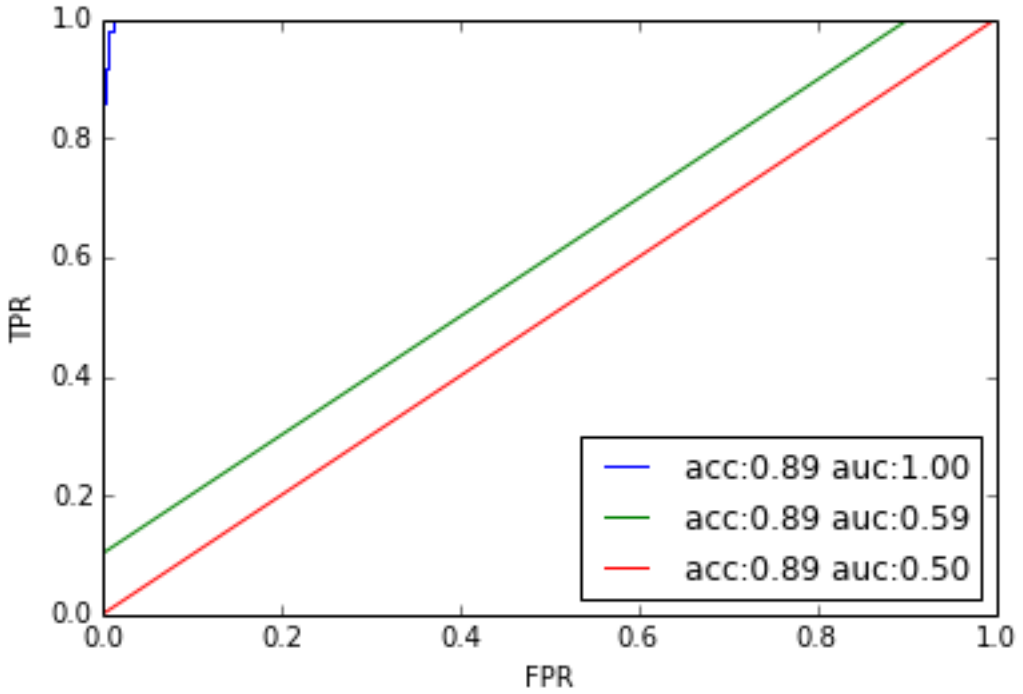
```
cross_val_score(SVC(), X_train, y_train)
>>> array([ 0.9,  0.9,  0.9])

cross_val_score(DummyClassifier("most_frequent"), X_train, y_train)
>>> array([ 0.9,  0.9,  0.9])
```

Scoring with imbalanced data

```
cross_val_score(SVC(), X_train, y_train)
>>> array([ 0.9,  0.9,  0.9])

cross_val_score(DummyClassifier("most_frequent"), X_train, y_train)
>>> array([ 0.9,  0.9,  0.9])

cross_val_score(SVC(), X_train, y_train, scoring="roc_auc")
>>> array([ 1,0,  1.0,  1,0])
```

Scoring with imbalanced data

```
cross_val_score(SVC(), X_train, y_train)
>>> array([ 0.9,  0.9,  0.9])

cross_val_score(DummyClassifier("most_frequent"), X_train, y_train)
>>> array([ 0.9,  0.9,  0.9])

cross_val_score(SVC(), X_train, y_train, scoring="roc_auc")
>>> array([ 1,0,  1.0,  1,0])
```

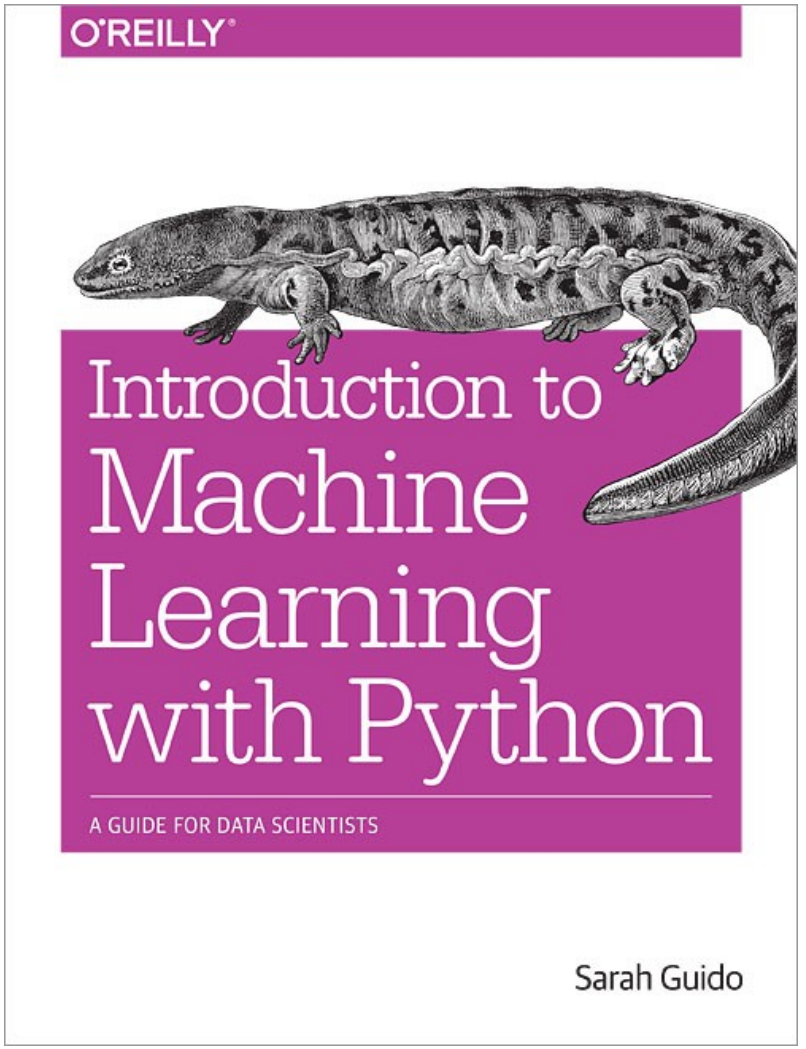


Video Series
Advanced Machine Learning with scikit-learn

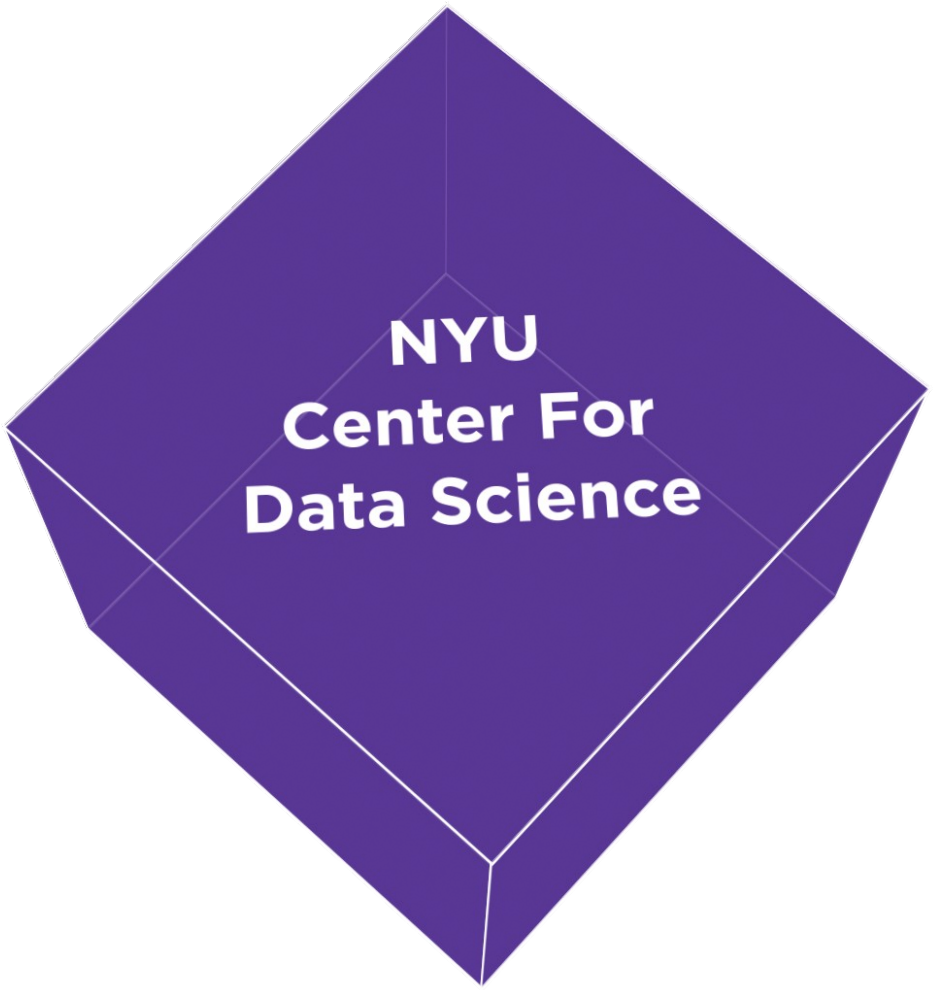
50% Off Coupon Code: AUTHD

Video Series
Advanced Machine Learning with scikit-learn

50% Off Coupon Code: AUTHD



CDS is hiring Research Engineers

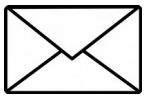


Work on your favorite data science open source project full time!

Thank you for your attention.


@t3kcit


@amueller


importamueller@gmail.com


http://amueller.github.io